# *DePICT - A Conceptual Model for Digital Preservation*

*Angela Dappert*

This thesis is submitted in partial fulfilment of the requirements of the award of the degree of Doctor of Philosophy of the University of Portsmouth

*January, 2013*

# *Abstract*

Digital Preservation addresses a significant threat to our cultural and economic foundation: the loss of access to valuable and, sometimes, unique information that is captured in digital form through obsolescence, deterioration or loss of information of how to access the contents. Digital Preservation has been defined as “The series of managed activities necessary to ensure continued access to digital materials for as long as necessary” (Jones, Beagrie, 2001/2008).

This thesis develops a conceptual model of the core concepts and constraints that appear in digital preservation - DePICT (Digital PreservatIon ConceptualisaTion). This includes a conceptual model of the digital preservation domain, a top-level vocabulary for the concepts in the model, an in-depth analysis of the role of digital object properties, characteristics, and the constraints that guide digital preservation processes, and of how properties, characteristics and constraints affect the interaction of digital preservation services. In addition, it presents a machine-interpretable XML representation of this conceptual model to support automated digital preservation tools.

Previous preservation models have focused on preserving technical properties of digital files. Such an approach limits the choices of preservation actions and does not fully reflect preservation activities in practice. Organisations consider properties that go beyond technical aspects and that encompass a wide range of factors that influence and guide preservation processes, including organisational, legal, and financial ones. Consequently, it is necessary to be able to handle ‘digital’ objects in a very wide sense, including abstract objects, such as intellectual entities and collections, in addition to the files and sets of files that create renditions of logical objects that are normally considered. In addition, we find that not only the digital objects' properties, but also the properties of the environments in which they exist, guide digital preservation processes.

Furthermore, organisations use risk-based analysis for their preservation strategies, policies and preservation planning. They combine information about risks with an understanding of actions that are expected to mitigate the risks. Risk and action specifications can be dependent on properties of the actions, as well as on properties of objects or environments which form the input and output of those actions. The model presented here supports this view explicitly. It links risks with the actions that mitigate them and expresses them in stakeholder specific constraints. Risk, actions and constraints are top-level entities in this model.

In addition, digital objects and environments are top-level entities on an equal level. Models that do not have this property limit the choice of preservation actions to ones that transform a file in order to mitigate a risk. Establishing environments as top-level entities enables us to treat risks to objects, environments, or a combination of both.

The DePICT model is the first conceptual model in the Digital Preservation domain that supports a comprehensive, whole life-cycle approach for dynamic, interacting preservation processes, rather than taking the customary and more limited view that is concerned with the management of digital objects once they are stored in a long-term repository.

# *Table of contents*

# *Table of figures*

# *Table of tables*

# *Declaration*

Whilst registered as a candidate for the above degree, I have not been registered for any other research award. The results and conclusions embodied in this thesis are the work of the named candidate and have not been submitted for any other academic award.

Word count: 65485

# *Acknowledgements*

I would like to thank all who contributed to the development of this thesis.

- The members of my supervision team in the School of Creative Technologies, Dr. Janet Delve and Prof. David Anderson, who encouraged me to turn my professional work in digital preservation into an academic product, who gave invaluable advice and support, and who more than succeeded in disproving my claim that getting a Ph.D. will always be a "nightmare experience";
- Dr. William Kilbride, Director of the Digital Preservation Coalition who saw value in my work for the digital preservation community and made it possible for me to complete this thesis while working for the Coalition;
- The European Commission who co-funded my research
    - under the Information Society Technologies (IST) Programme of the 6th Framework Programme for research and technological development and demonstration under grant agreement IST-033789, the Planets project, a four-year project to address core digital preservation challenges and
    - under the Information and Communication Technologies (ICT) Programme of the 7th Framework Programme for research and technological development and demonstration activities under grant agreement ICT-270137, the SCAPE project, a three-and-a-half-year project to address scalable digital preservation;
    - under the Information and Communication Technologies (ICT) Programme of the 7th Framework Programme for research and technological development and demonstration activities under grant agreement ICT-269940, the TIMBUS project, a three-year project to address digital process preservation;
- My colleagues at the British Library, at the Digital Preservation Coalition, on the PREMIS Editorial Committee and on the Planets, SCAPE and TIMBUS projects, who share my passion for the field and help me clarify my understanding in countless discussions;
- Dr. Brett Stevens and Dr. Steve Hand for reviewing the thesis and providing valuable feedback;
- My husband, Dr. Adam Farquhar, who encouraged me.

# *Dissemination*


Dappert, A., Ballaux, B., Mayr, M., van Bussel, S. (2008). Report on policy and strategy models for libraries; archives and data centres. Planets report PP2-D2. Retrieved from http://www.planets-project.eu/docs/reports/Planets_PP2_D2_ReportOnPolicyAndStrategyModelsM24_Ext.pdf, accessed on 23 November 2012

Dappert, A. (2009). Report on the Conceptual Aspects of Preservation, Based on Policy and Strategy Models for Libraries, Archives and Data Centres. PLANETS report PP2-D3. Retrieved from http://www.planets-project.eu/docs/reports/Planets_PP2_D3_ReportOnPolicyAndStrategyModelsM36_Ext.pdf, accessed on 23 November 2012

Dappert, A., Farquhar, A. (2009a). Modelling Organizational Preservation Goals to Guide Digital Preservation. International Journal of Digital Curation, 4(2), 119-134. doi:10.2218/ijdc.v4i2.102.Retrieved from http://www.ijdc.net/index.php/ijdc/article/download/123/126, accessed on 23 November 2012

Dappert, A., Farquhar, A. (2009b). Significance Is in the Eye of the Stakeholder. (M. Agosti & et alii, Eds.) ECDL'09 Proceedings of the 13th European conference on Research and advanced technology for digital libraries, September/October 2009, LNCS 5714, 297-308. Berlin, Heidelberg: ©Springer Verlag. Retrieved from http://www.planets-project.eu/docs/papers/Dappert_SignificantCharacteristics_ECDL2009.pdf or http://www.bl.uk/aboutus/stratpolprog/ccare/pubs/2009/ipres2009-Dappert%20and%20Farquhar.pdf, accessed on 8 May 2012

Dappert, A. (2010). Deal With Conflict, Capture the Relationship: The Case of Digital Object Properties. iPRES 2010: The Seventh International Conference on Preservation of Digital Objects Retrieved from http://www.ifs.tuwien.ac.at/dp/ipres2010/papers/dappert-05.pdf, accessed on 23 November 2012

Dappert, A., Enders, M. (2010). Digital Preservation Metadata Standards, NISO Information Standards Quarterly. 22(2), June 2010. Retrieved from http://www.loc.gov/standards/premis/FE_Dappert_Enders_MetadataStds_isqv22no2.pdf accessed on 10 August 2012

Dappert, A., Farquhar, A. (2011). Implementing Metadata that Guide Digital Preservation Services. International Journal of Digital Curation, 6(1). Retrieved from http://ijdc.net/index.php/ijdc/article/view/176, accessed on 30 November 2012

Dappert, A. (2011). Risk Assessment of Digital Holdings. Retrieved from http://www.digitalpreservationsummit.de/presentations/dappert.pdf, accessed on 17 August 2012

Dappert, A., Jackson, A., Kimura, A. (2011). Developing a Robust Migration Workflow for Preserving and Curating Hand-held Media, iPres 2011, The Eighth International Conference on Preservation of Digital Objects. Retrieved from http://ipres2011.sg/conference-procedings#, accessed on 16 November 2012

Dappert, A., Peyrard, S., Delve, J., Chou, C. (2012). Describing Digital Object Environments in PREMIS. iPRES 2012: The Ninth International Conference on Preservation of Digital Objects. Retrieved from https://ipres.ischool.utoronto.ca/sites/ipres.ischool.utoronto.ca/files/iPres%202012%20Conference%20Proceedings%20Final.pdf, accessed on 23 November 2012 2012

Dappert, A. (2012). Proposed Data Model Changes for PREMIS 3.0. (PREMIS Implementation Fair). 2 October 2012. Presentation slides. Retrieved from http://timbusproject.net/component/docman/doc_download/73-proposed-data-model-changes-for-premis-30, accessed on 14 December 2012

# *Terminology*

Technical terminology is rendered

- in Quote style if it is defined in external work,
- in `Code style` if it is defined in the DePICT model.

It is important to note that the terminology around characteristics, properties, values, etc. throughout the preservation literature is very inconsistent. The literature, for example, refers to significant properties just as it refers to essential characteristics. Effort was made to ensure that the terminology in this thesis is internally consistent while unifying the use with other work wherever possible. The source of the definition in the tables below is given in parentheses. Many examples for and explanations of the terms are contained in chapter 3 on the conceptual model. Some key terms are defined in the following.

**References**

**Table 1: Frequently referenced related work**

| Term | Definition |
|---|---|
| DePICT | **D**igital **P**reservat**I**on **C**onceptualisa**T**ion |
| Planets | A four-year project co-funded by the European Union 2006 – 2010 to address core digital preservation challenges. (Farquhar, Hockx-Yu, 2007; Planets, nd) |
| SCAPE | A three-and-a-half-year project co-funded by the European Union 2011 – 2014 to address core digital preservation challenges. (Edelstein et al., 2011; SCAPE, nd) |
| TIMBUS | A three-year project co-funded by the European Union 2011 – 2014 to address digital preservation of business processes. (Edelstein et al., 2011; TIMBUS, nd) |
| PREMIS | The *de facto* standard on digital preservation metadata. A data dictionary with associated optional XML and RDF implementations. (PREMIS, 2012) |
| OAIS | The ISO Reference Model for an Open Archival Information System (OAIS) which preserves digital assets and makes them available (CCSDS, 2012). |

## General concepts

**Table 2 General digital preservation terminology**

| Term | Definition |
|---|---|
| Digital Preservation | Digital preservation combines policies, strategies and actions that ensure access to digital content over time. (ALA, for a more detailed definition please follow the link in the bibliography (PARS, 2007).)<br><br>This model limits the scope to preservation aspects that maintain digital objects that are at preservation risk by mitigating those risks through preservation actions. Preservation constraints are used to determine the presence of those risks and to guide the choice of acceptable preservation actions. |
| Preservation Policy | A formal statement of direction or guidance as to how an organisation will carry out its preservation mandate, functions or activities, motivated by determined interests or programs. (based on InterPARES2 (InterPARES, nd)) |
| Preservation Strategy | The strategy is a procedure of preservation actions to preserve a collection of digital objects. The preservation strategy thus contains a detailed description of the preservation action(s) to be taken, including<br>• used hardware and software,<br>• parameter settings for used tools and actions, and<br>• input and output file format, and<br>• available metadata about the action(s).<br><br>(Adapted from the Planets project internal definition as of 2009-06-01) |

**Entities in the conceptual model**

**Table 3 Terminology in support of a digital preservation conceptual model**

| Term | Definition |
|---|---|
| *Bitstream* | A *Bitstream* is contiguous or non-contiguous data within one or more *Files* that has meaningful common properties for preservation purposes. |
| *Bytestream* | An ordered sequence of bytes. A file is a special *Bytestream* |
| *Characteristic* | A *Characteristic* of an entity is the concrete *Value* which this entity has for an abstract *Property* in a defined context (a concrete *Property/Value* pair).<br>In the model it is the *Characteristic* of a *PreservationObject*, *Environment* or *PreservationAction*. |
| *Constraint* | A limitation or restriction on the space of allowable *Preservation-Actions*. |
| *Environment* | A set of factors which constrain a *PreservationObject* or *PreservationAction* and that are necessary to interpret it. |
| *File* | A *File* is a named and ordered sequence of bytes that is known by an operating system. A file can be zero or more bytes and has a file format, access permissions, and file system characteristics such as size and last modification date (PREMIS, 2012). |
| *IntellectualEntity* | A set of content that is considered a single intellectual unit for purposes of management and description; a distinct intellectual or artistic creation that is considered relevant, by curatorial decision, to a Designated Community in the digital preservation context. (adapted from PREMIS (PREMIS, 2012)) |
| *PreservationService* | A *PreservationService* is an *Agent* that provides a core service supporting the goal of digital preservation. |

| Term | Definition |
|---|---|
| *PreservationAction* | The execution of a *PreservationService* that mitigates a *PreservationRisk* to the continued viability, renderability, understandability, and authenticity of a *PreservationObject* across time and changing *Environments*. It ensures the satisfaction of their *Constraints*. A *TransformationPreservationAction* may transform the *PreservationObject* itself, the *Environment* required to support access to the *PreservationObject*, or a combination thereof.<br><br>A *PreservationAction* is an *Event* resulting from the execution of a *PreservationService*. |
| *Policy* | Representations that specify *Constraints* that make a stakeholder's values, priorities or goals explicit and influence a preservation process. |
| *PreservationObject* | A *PreservationObject* is any object that can directly or indirectly be at risk and needs to be digitally preserved. |
| *PreservationRisk* | A *PreservationRisk* arises when a *Characteristic* of a *PreservationObject* or of an *Environment* of a *PreservationObject* conflicts with the stakeholder's *RiskSpecifyingConstraints*.[1] |
| *Property* | An abstract attribute, trait or peculiarity suitable for describing *PreservationObjects*, *PreservationActions* or *Environments*. |
| *Representation* | One physical embodiment of an *IntellectualEntity*.<br><br>The set of *RepresentationBitstreams* that are needed to create one rendition of an *IntellectualEntity* together with the necessary structural information. |
| *RepresentationBitstream* | *RepresentationBitstreams* are the logical, ideal bitstreams that make up the *Representation*. |

[1] This thesis does not distinguish between risks (things that may happen) and issues (things that have happened). Nor does it distinguish between threats and opportunities. In consequence *Preservation-Actions* include proactive and reactive actions.

| Term | Definition |
|---|---|
| SignificantProperty<br>/<br>*SignificanceConstraint* | The characteristics of digital objects that must be preserved over time in order to ensure the continued accessibility, usability, and meaning of the objects, and their capacity to be accepted as evidence of what they purport to record. (Wilson, 2007)<br>*Constraints* in a specific context, expressing a combination of *Characteristics* of *PreservationObjects* or *Environments* that must be preserved or attained in order to ensure the continued accessibility, usability, and meaning of *PreservationObjects*, and their capacity to be accepted as evidence of what they purport to record. |
| *Value* | Every *Characteristic* has a *Value* which can either be assigned or be inherent in the object. The *Value* can be looked up if it is stored explicitly or measured with an associated measuring tool, or deduced with a given logic if it is inherent in the object. |
| *ValueOrigin* | The *ValueOrigin* entity provides a way to specify where a specific *Value* comes from or how it can be obtained. There can be multiple ways of obtaining the *Value* of a *Property* that do not conflict, measured by a different technique, using a different tool, or by a different agent. |

# 1 Introduction

For any field of information management, to solve its relevant problems in an effective and collaborative way it is essential to have a shared, thorough understanding of its domain. This includes a shared terminology and shared models, both of key conceptual entities and their properties, and of the functions required to execute its tasks. Based on them, it is possible to build a set of successful end-to-end services. A shared notion of the data and metadata that need to be exchanged between services is essential in order to correctly put together the pieces, and to be able to implement the entire functionality. Additionally, this metadata must be able to capture the goals and constraints of the stakeholder who undertakes them.

This is particularly true in an evolving field, such as digital preservation, that is sufficiently young that a shared conceptual understanding has not evolved to a significant point. This thesis is developing such a conceptual model.

## 1.1 Digital preservation

Our society has eagerly embraced the move from traditional information processing, mostly on paper, to digital information processing, benefiting from its greatly improved functionality. Remote access, full-text search, easy copying, linking documents or mixed-media, dynamic execution of actions or simulations, compact storage, ability to edit, and improved collaborative generation of information objects to name but a few. The different natures of digital data carriers, digital information encoding and access to digital information objects bring, however, new threats to the long-term preservation of those information objects in digital form.

Digital preservation is about mitigating those risks. According to the American Library Association (ALA) (PARS, 2007) the field of digital preservation 'combines policies, strategies and actions that ensure access to digital content over time'. Jones and Beagrie (Jones, Beagrie, 2001/2008) define it as 'the series of managed activities necessary to ensure continued access to digital materials for as long as necessary'. An early analysis of the field by the Task Force on Archiving of Digital Information can be found in the report by Garrett et al. (1996).

Digital repositories are computer systems that ingest, store, manage, preserve, and provide access to digital content for the long-term. This requires them to go beyond simple file or bitstream preservation. They must focus on preserving the information *per se* and not just the current file-based representation of this information. It is the actual information content of a document, data-set, or sound or video recording that should be preserved, not the Microsoft

Word file, the EXCEL spreadsheet or the Quicktime movie. The latter represent the information content in a specific file format that will become obsolete in the future.

The duration of the required accessibility varies from stakeholder to stakeholder, but may be indefinite. Memory institutions such as libraries, archives and museums have been leading the effort, since they are charged with indefinite preservation of their societies' artefacts. They are joined by regulated industry sectors, such as aircraft design and manufacture, and pharmaceuticals. Increased involvement of broader industry sectors in digital preservation research is a sign that awareness of the need for preserving digital objects over the long-term is spreading from the traditional champions in memory institutions and heavily regulated private sectors to the general private sector.

### *1.1.1 Digital preservation risks*

Valuable scientific and cultural information assets are created, stored, managed and accessed digitally, but the threat of losing them over the long term is high. If insufficient care is taken, any data and information management activity poses a threat, for example when digital objects are copied, moved, renamed or reformatted. Digital media are brittle: they decay and are short lived. Over time, changes in the external environment pose additional risks: data carrier and reader technology become obsolete; software and hardware technologies required to access them continue to evolve at a rapid rate and fall into obsolescence; formats that are used to represent digital objects fall into disuse; "representation information" that specifies how to access or interpret them is lost; especially, digital material encoded in proprietary formats becomes inaccessible when the associated software is no longer available since there is no open specification that would permit reconstruction of the rendering software; changes in organisations' cultural, and financial priorities add risk to continued accessibility and long-term preservation of our digital assets. Unlike print-based materials, digital assets cannot survive significant gaps in preservation care. Figure 1 illustrates how various risk sources (on the left) can be matched to objects and environment components that are potentially affected by them (on the right).

These risks[2] are in addition to the customary day-to-day risks encountered in information management. Data safety and security obviously need to be guaranteed in the short-term in order to guarantee availability in the long run. In addition, techniques, such as replication on

[2] This thesis does not distinguish between risks (things that may happen) and issues (things that have happened). Nor does it distinguish between threats and opportunities. In consequence *Preservation-Actions* include proactive and reactive actions.

geographically distant systems, backups, hash-functions that allow the repository to determine whether corruptions have happened to a file, and digital signatures are appropriate. It is advisable that a variety of data carriers is employed in order to reduce dependence on a single storage technology. Employing open standards for encoding, file systems, file formats and metadata representation increases our ability to use digital assets now and in the future. This has to be combined with business continuity and disaster response techniques (Burtles, 2007) in order to respond to sudden risks to the assets' availability. These latter concerns are customarily not included in the scope of digital preservation.

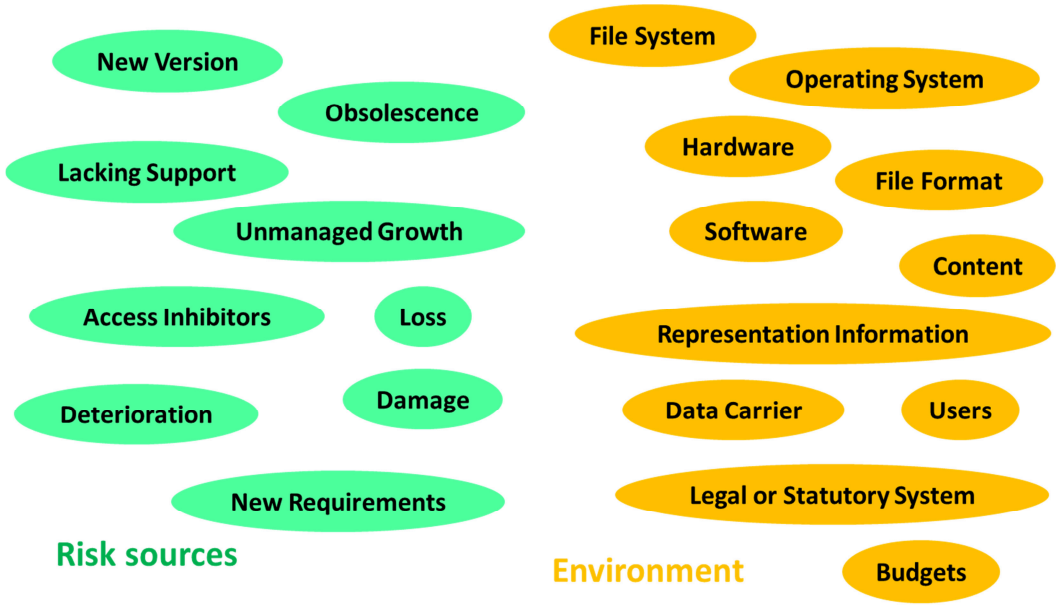



**Figure 1: Relating preservation risk sources to potentially affected objects and environments**

### *1.1.2 Digital preservation goals*

Digital Preservation goals are to ensure that

- digital content is within the physical control of the repository;
- digital content can be uniquely and persistently identified and retrieved in future;
- sufficient information is available so that digital content can be understood by its designated user community (representation information);
- significant characteristics of the digital assets are preserved even as data carriers or physical representations change;
- physical media are cared for and corruption is detected and repaired (fixity);
- digital objects remain renderable or executable;

- digital objects remain whole and unimpaired and that it is clear how all the parts relate to each other (integrity); and
- digital objects are what they purport to be (authenticity).

This is illustrated in Figure 2 and discussed in depth in Caplan (2008).

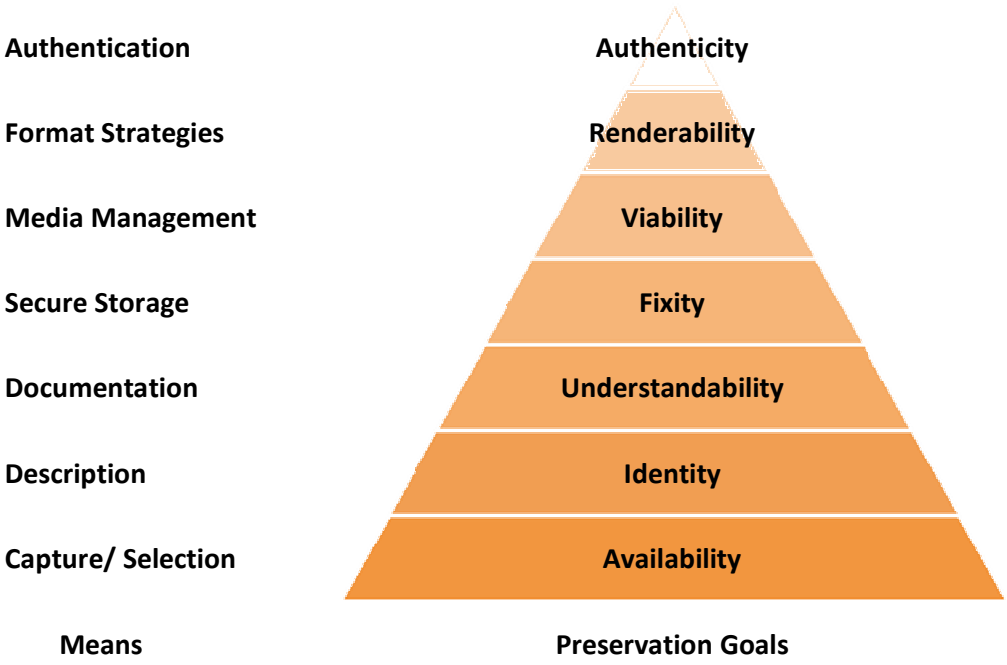



**Figure 2: Preservation Goals (Caplan, 2008)**

### *1.1.3 Digital preservation activities*

Digital preservation activities are necessary during the whole life-cycle of digital information objects. At their creation, thought should be given to their resilience, long-term documentation and self-documentation. The file format should be selected not only based on its functional characteristics, but also based on how supportive it is of digital preservation goals, and should fit the individual organisation's use and preservation needs. Preservation metadata should be gathered and represented in a way that optimally supports future preservation actions. During daily management, long-term preservation needs to be considered to ensure, for example that lossy conversions only happen with the full intention of their curators or that any relevant change that would impact the authenticity of the original object is documented so that the object is accompanied by a trail of provenance metadata that provides accounts to future users as to what degree this object reflects the original that it purports to be. In the long-term, the object itself and the environment on which it depends for its execution need to be monitored for corruption or

obsolescence; and mitigating actions have to be taken to ensure continued accessibility. It is important to note that digital preservation is not something that is only applied in the distant future, where obsolescence may occur. Any day-to-day manipulation of digital assets can lead to loss of accessibility in the long run if digital preservation principles are not applied at all times.

Preservation services (Lee et al., 2012; DCC, nd; Hitchcock et al., 2010) are being developed by the digital preservation research community and private sector organisations to support digital curators in their stewardship task. These preservation services work together towards that goal, and the hope is that we might develop an interoperable tool kit to support the curators' decision making with knowledge-rich decision support systems, to automate preservation tasks, wherever human intervention is not needed, and that file and storage systems and file formats become inherently easier to preserve, standards based, less volatile and longer-lived. Preservation services go beyond the execution of preservation actions that mitigate a risk by transforming the digital object and its environment to a state where the risk no longer applies. They, for example, comprise the description of digital assets in order to aid their long-term management, discovery and retrieval; they include: automatic characterisation to determine technical and other properties of digital assets that in turn help to manage them; preservation monitoring to detect the presence of risk; planning for preservation actions; and validation in which one assesses whether all the significant characteristics of a digital object are preserved after the execution of a preservation action. In chapter 4, the creation of a model of interacting preservation services is discussed, and background information on existing tools and services is considered within that context.

The main focus in digital preservation debates is, however, often on the actual preservation actions taken that directly mitigate an existing risk and the methodologies applied.

### *1.1.4 Digital preservation methodologies*

Digital preservation professionals have a choice of preservation methodologies and need to decide in their preservation planning task which methodology best mitigates the preservation risk they are addressing. This is not always a straight-forward choice. It is determined by their preservation requirements, resources, organisational guidelines, skill sets and availability of the necessary tools and the nature of risk that is to be mitigated. In the past there were fundamental arguments for or against certain choices of methodology as a matter of principle, especially between migration and emulation (Bearman, 1999; Rothenberg, 1999; Stawowczyk Long, Pearson, 2009). But, since each methodology addresses specific preservation risks and has specific strengths and weaknesses, it is increasingly understood that these are complementary

methodologies (Anderson, Delve, Pinchbeck, 2010) and that their choice is context dependent (Dappert, Farquhar, 2009b; Becker, Kulovits, Guttenbrunner, Strodl, 2009) and depends on the requirements given in the particular context. It is important for an organisation to define a clear preservation policy so that it can develop the most appropriate preservation strategy with the right choices of preservation methodologies.

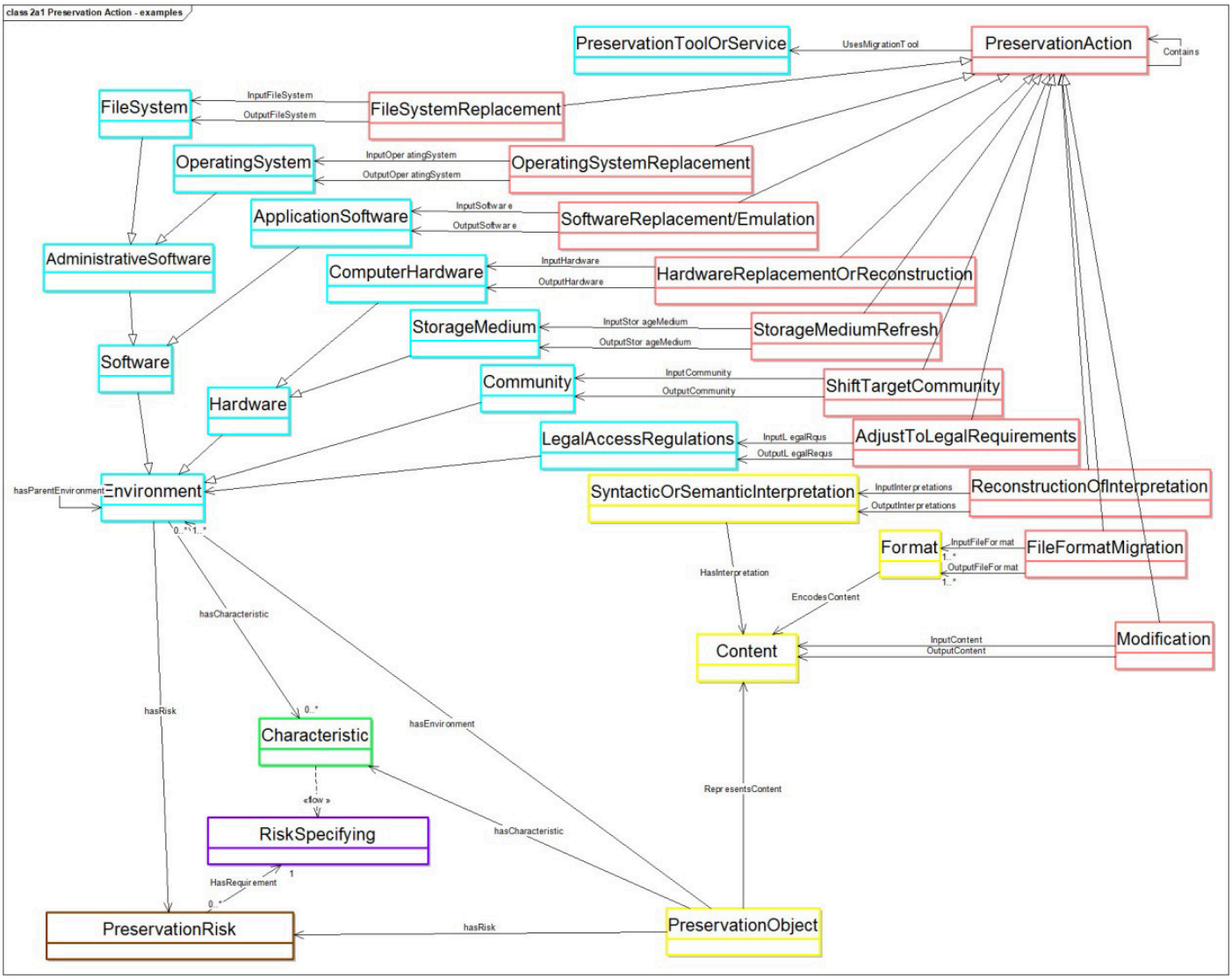



**Figure 3: Example preservation methodologies matched to sub-classes of *PreservationObjects* or *Environments* and *PreservationRisk* (Dappert, Farquhar, 2009a)**

Chue Hong et al., (2010) suggest seven different methodological options for preservation and sustainability of software in particular:

- Technical preservation (techno-centric) - Preserve original hardware and software in the same state as they are now;

- Emulation (data-centric) - Emulate the original hardware / operating environment, while keeping the software in the same state;
- Migration (functionality-centric) - Update the software as required to maintain the same functionality, by porting or transferring it;
- Cultivation (process-centric) - Keep software 'alive' by moving to a more open development model, bringing on board additional contributors and spreading knowledge about it;
- Hibernation (knowledge-centric) - Preserve the knowledge of how to resuscitate/recreate the exact functionality of the software at a later date;
- Deprecation - Formally retire the software without leaving the option of resuscitation/recreation;
- Procrastination - Do nothing.

As stated above, none of these is inherently preferable, but depend on the individual preservation situation. These specific methodologies (in adapted form) also apply to digital objects that are not software. But, in fact, in an holistic conceptual model, preservation methodologies can be interpreted in an even broader way, as illustrated in Figure 3, which matches example preservation risks against a wide choice of example preservation methodologies. They can include forms of bit preservation, repair, forensics, recovery, and reconstruction. They can also include back-ups onto paper or micro fiche. And they can include an intentional decision to no longer preserve the digital object, or to wait and see, in the expectation that better technology will come along. The most common preservation methodologies are discussed in some detail below. All methodologies are supported by the DePICT model presented in this thesis.

#### *1.1.4.1 Bit preservation / storage medium refresh and replication*

The bits that represent digital information are stored on various types of data carriers (or storage media). Data carriers can be physically damaged. Additionally they are subject to bit rot, a naturally occurring reversal of some bits' values that is caused by ageing of the data carrier. The obvious methodology for mitigating this risk is bit preservation, the process of copying the data carrier content bit-by-bit onto newer data carriers. This is also called storage medium refresh. In order to be able to recover from bit loss, it is additionally necessary to hold replicas remotely in order to address larger scale disasters and to create backups from which one could recover damaged digital objects locally. These replicas and backups are also created through bit copying.

The bit preservation strategies that are chosen depend on preservation requirements and vary for different digital materials. Confidentiality, bit safety, availability and cost requirements influence the preservation strategies (Zierau, Kejser, Kulovits, 2010). Data carrier properties influence data carrier choice, and failure rates influence bit preservation frequency (Rosenthal, 2010). Bits can be copied either as data carrier "image" in the form in which it was stored on the data carrier or by copying the bits of the files stored on the data carrier independent of the data carrier encoding format (Woods, Brown, 2008; Dappert, Jackson, Kimura, 2011). Bit preservation can go beyond mere storage medium refresh. It can include a move from an obsolete carrier type to a better supported carrier type. In that case it is often natural to also move away from the old carrier's encoding format.

Bit preservation is validated through fixity checking, in which the hash-sums of each file before and after copying are compared for identity to ensure that the file was not damaged during copying (Novak, 2006). Additionally, integrity checking ensures that all files have been copied.

The LOCKSS program (LOCKSS, nd) uses bit preservation as a preservation methodology by keeping several copies decentralized and distributed at their membership organisations, while giving them local custody and control of their assets.

#### *1.1.4.2 Migration*

Migration[3] (Garrett et al., 1996) is a preservation methodology in which digital objects are translated from one format to another. Migration happens proactively, when the steward of digital objects decides that all ingested objects should be "normalized", meaning that they are transformed to a smaller set of preferable formats that are supported in the repository. It also happens reactively, if, during preservation watch, it is found that the current format of a digital object is no longer sufficiently supported. The digital object is then translated into a better supported format. This can be a translation from one file format to another; it can also be a translation of higher-level encodings, such as a port of a software program from one programming language to another.

In this form of preservation methodology the bits are not preserved, but the contents of the digital objects should be. In practice, however, different representation formats have different properties and capabilities of representing digital object content. Because of this it is often inevitable that during migration some of the characteristics of the original object are lost in

[3] ISO 13008:2012 (2012) distinguishes "conversion" and "migration", where a conversion is a shift in format, and migration is a shift from data carrier to data carrier; but this distinction is not typically observed in the digital preservation parlance.

translation. For example, when migrating a document from Microsoft Word to Adobe PDF the editing history available in Word is lost in the resulting PDF. In practice, the programmes that perform the migration are also not always correctly implemented or lose characteristics of the object in the translation (Kulovits et al., 2009). Migrations are validated by ensuring that the "Significant Characteristics", those characteristics that are deemed essential by the stakeholders are preserved (Thaller, M., et al., 2008; Knight, 2008; Knight, Pennock, 2009; Dappert, Farquhar, 2009b).

#### *1.1.4.3 Emulation*

Emulation (Rothenberg, 1998) is a preservation methodology in which the digital object remains (largely) unchanged, but the platform on which it is to be rendered or executed is brought up to date. Since an obsolete digital object cannot directly be used on a more modern platform, a piece of software, called an emulator has to be written, which recreates the original computer environment on the more modern platform. Emulators can be written for individual software applications, operating systems, or hardware platforms. For dynamic, digital objects, such as computer software, emulation is often the most cost-effective preservation methodology. All software that ran on the old platform can run on the newer one using the same emulator. Were the software to be migrated (ported) instead, one would have to migrate every piece of software that should be kept usable. Nonetheless, emulation is a very complex (Kuchera, 2011; Fayzullin, M., 1997-2000) and expensive task. But the most prominent obstacle to emulation is often found in potential copyright violations of preserving proprietary software or hardware (Anderson, 2011; Charlesworth, 2012).

A Universal Virtual Computer (UVC) (Lorie, 2001) is a program which contains a set of computer instructions. At any given time one can implement this openly specified UVC on the currently available computer platforms. Computer programs that have been written in the past to run on a UVC can in this way run on the current platforms. This strategy is a combination of emulation and migration. The underlying idea is that it will be simpler in the future to create an emulation program for the UVC than it would be to emulate a complex platform. UVCs have to be considered a research product, at this point; they are not available for practical use.

Validating emulation is difficult (Guttenbrunner, Rauber, 2012), since it is often used to preserve dynamic digital objects and it is not easy to validate that every aspect of the original behaviour has been preserved in the emulated performance. Guttenbrunner, Wieners, Rauber and Thaller. (2010) use snapshots of the emulated and original process to validate the authenticity of the emulation.

Example projects for developing emulation solutions are the KEEP project (KEEP, 2012) which has developed an emulation framework and the Dioscuri project (van der Hoeven, 2007).

#### *1.1.4.4 Computer museums*

Computer museums address digital preservation on a physical level in which software and hardware are preserved in their historical condition (Anderson, Delve, Powell, 2012; Ainsworth, Avram, Sheard, 2010; Demant, 2010). Preserved storage equipment in its native environment can be used to retrieve data from obsolete data carriers, often the only way to access this information, unless newer recovery technologies exist (see section 1.1.4.5). Original platforms are also the only way to validate whether an emulation implementation preserves the significant characteristics of the emulated system. "It is crucial to establish if, for example, some unexpected behaviour exhibited by a digital object is the result of a defect introduced during preservation or was originally present" (Anderson, Delve, Pinchbeck 2010). They further state, that while it is important to carefully document platform characteristics in order to use this knowledge later on, for example, when building an emulator, it is a fact that all platforms have undocumented or un-documentable features. These features can be investigated as long as the original platform is physically maintained.

The great advantage of this approach is that the rendering or execution of obsolete digital objects is as authentic to the original one as possible, barring inevitable changes to the original environment that cannot be reproduced. A short-coming of this approach is that it offers limited regional access and that parts eventually might not be replaceable. For example, there are only enough tape heads in existence to read only a small part of the ¾ inch tape collection at the British Library.

#### *1.1.4.5 Forensics, recovery and reconstruction*

Bit preservation, migration, emulation and platform preservation are methodologies that are applied while the knowledge necessary to execute them is still at hand, even if this may be done with difficulty, as often is the case, especially in emulation. Some other preservation methodologies however are reactive. Damage has already happened or information necessary has been lost.

Digital Forensics originated in law enforcement's need to investigate digital objects of an unknown nature that are suspected of being able to provide evidence. Its methodologies for dealing with issues of data recovery and legacy formats of applications, hardware, file systems and operating systems, while ensuring authenticity, security and privacy, are equally applicable to

personal digital archives. Personal digital archives that have been accepted into the care of a collection without sufficient information about their technical properties, their content, the presence of possibly sensitive information and their creation history. Kirschenbaum, Ovenden and Redwine (2010) state:

> *"The same forensics software that indexes a criminal suspect's hard drive allows the archivist to prepare a comprehensive manifest of the electronic files a donor has turned over for accession; the same hardware that allows the forensics investigator to create an algorithmically authenticated "image" of a file system allows the archivist to ensure the integrity of digital content once captured from its source media; the same data-recovery procedures that allow the specialist to discover, recover, and present as trial evidence an "erased" file may allow a scholar to reconstruct a lost or inadvertently deleted version of an electronic manuscript"*

Recovery mechanisms are applied when the original computing environment is no longer available, inferior to newer methodologies or would risk further degradation to old data carriers. For example, recovery mechanisms have been developed to capture sound content from old wax cylinders and from LP records using a confocal microscope. By using optical readers that encode the sound directly from the optical scan into digital form without first recreating and recording the sound waves, the new technology may even enhance the sound compared to the original stylus play-back mechanism. This methodology also supports the reconstruction of damaged or broken cylinders (Fadeyev, Haber, 2003).

Reconstruction may be needed when data or data carriers have been partially damaged. This is the case when data can only be partially recovered from damaged data carriers or if copying errors have occurred. Software that identifies and repairs file damage is, for example, being developed in projects, such as AQuA (2011).

## 1.2 Research goal - a conceptual model for digital preservation - DePICT

For any field of information management to solve its relevant problems in an effective and collaborative way, it is essential to have a shared, thorough understanding of its domain. This includes a shared terminology and shared models, both of key conceptual entities and their properties and of the functions required to execute its tasks. Based on them it is possible to build a set of successful end-to-end services. A shared notion of the data and metadata that need to be exchanged between services is essential in order to correctly put together the pieces and to be able to implement the entire functionality that is needed. Additionally this metadata must be able to capture the strategy and policy goals and constraints of the stakeholder who undertakes them.

This is particularly true in an evolving field, such as digital preservation, that is sufficiently young that a shared conceptual understanding has not evolved to a significant point.

As the field matures the need for the existence of such a model is apparent; not having it now risks that incompatible and partial approaches solidify. In the early days of digital preservation, there was considerable work devoted to establishing conceptual models (e.g., Rothenberg, 1998). These approaches, however, did not benefit from practical experience dealing with actual digital material at scale. Today, practice at leading institutions provides the essential experience that can guide the development of a fruitful model.

### 1.2.1 Gaps - the need for a comprehensive conceptual model for digital preservation

Existing conceptual and functional modelling approaches in the field of Digital Preservation will be discussed in the next chapter. For each approach the analysis will show how gaps in those models and approaches prevent end-to-end life-cycle modelling or leave stakeholder needs unsatisfied. To motivate the research goal, the key limitations of conceptual models currently in use are summarised here:

- They tend to focus on statically recording characteristics and events, rather than on dynamically supporting preservation processes. For example. PREMIS (2012), intentionally, only describes information and data objects stored in an Open Archival Information System (CCSDS, 2012) repository, rather than the interaction of digital preservation services and the whole lifecycle.

- They focus only on technical constraints, rather than considering the overall context. Digital preservation activities can only succeed if they go beyond the technical properties of digital objects. They must consider properties that encompass a wide range of factors that influence and guide preservation processes, including organisational, regulatory, legal, and financial ones that are captured as strategy and policy goals and constraints of the institution that undertakes them. They must take into account the cultural and institutional framework in which data, documents and records are preserved. Furthermore, because organisations differ in many ways, a one-size-fits-all approach cannot be appropriate.

- They limit themselves to certain types of digital objects, for example, to files rather than sets of files which create renditions of logical objects, abstract objects, such as intellectual entities and collections, or complete rendering stacks. In the simplest case, preservation of files, however, also requires the preservation of the representation information that is necessary to understand the preservation objects in the future. Even in the simplest case this representation information is much more complex than simple files, and requires the ability to preserve complex objects.

- They focus on particular solution approaches, such as migration or emulation, exclusively. An approach that, for example, focuses on preserving technical properties of digital files limits the choices of preservation actions and does not fully reflect preservation activities. In practice, we find that not only the digital objects' properties, but also the properties of the environments in which they exist, guide digital preservation processes. Therefore, it is necessary that preservation objects and environments are top-level entities on an equal level. Models that do not have this property limit the choice of preservation actions to ones that transform a file in order to mitigate a risk. Establishing environments as top-level entities enables us to treat preservation risks to preservation objects, environments, or a combination of them.

- They describe functional interactions at a high level. For example, OAIS (CCSDS, 2012) does not sufficiently support modelling of interacting preservation services to capture the necessary information exchange.

- They tend to describe absolute solutions, rather than making them relative to the actual risk and the organisation’s goals, as expressed in their policies. Organisations use risk-based analysis (e.g. Drambora (McHugh, Innocenti, Ross, 2008)) for their preservation strategies, policies and preservation planning. They combine information about risks with

an understanding of actions that are expected to mitigate the risk. Risk and actions specifications can be dependent on properties of the preservation actions, as well as on properties of preservation objects or environments which form the input and output of those preservation actions. The model presented in this thesis supports this view explicitly. It links risks with the preservation actions that mitigate them and expresses them in stakeholder specific constraints. Risks, actions and constraints are top-level entities in this model.

These partial models have led to the development of practical solutions that do not match up and that ignore important aspects of the domain that need to be considered.

### *1.2.2 The research goal*

The overall aim of this thesis is to produce a comprehensive conceptual model for the field of digital preservation for expressing its core concepts, their relationships and requirements - DePICT (**D**igital **P**reservat**I**on **C**onceptualisa**T**ion). It must incorporate all relevant organisational characteristics and strategic directions, and cover the full life cycle of digital information objects from the moment of creation. It must define a high-level specific vocabulary that institutions can reuse for expressing their own policies and strategies and describing their processes and collections. In addition to providing a conceptual model and vocabulary, DePICT should support automated preservation services through an XML representation.

The existence of an overarching conceptual model that is not subject to the above limitations would mean that

- it can be shared by institutions and software applications to improve the exchange and the interoperability of data, metadata and software.
- it can provide a standard which can serve as a convenient starting point for creating individualised models for an institution, saving them time and helping avoid errors. This holds true even if the institution does not require a machine-interpretable specification. Institutions can reuse the high-level specific vocabulary for expressing their own policies and strategies and describing their processes.
- it can be used to describe preservation metadata for individual institutions, possibly, but not necessarily, in a machine-interpretable form, that guide preservation actions. This, in turn, enables preservation services and decision support to be based on organisational policy and strategy constraints.

- it adds to the scientific understanding of digital preservation.

This conceptual model must be suitable for

- modelling a very wide range of preservation services, such as risk monitoring; determining characteristics of objects, environments and tools; comparison of characteristics to determine authenticity; evaluation of candidate preservation actions; and evaluation and validation of preservation execution,
- modelling organisational as well as technical properties,
- modelling a very wide range of preservation methodologies from emulation, virtualisation, migration, recreation and bit-preservation to more abstract but equally important actions that update to current compliance or use requirements,
- modelling a very wide range of entities from logical to physical entities, including actions and environments,
- basing preservation actions on risk management by lining up preservation actions against the risks they mitigate,
- covering the full life cycle of digital information objects.

The resulting conceptual model should be a simple yet expressive representation of the digital preservation domain.

In particular, the research outputs of this thesis are

- a conceptual model of the digital preservation domain, based on domain requirements (see chapter 3).
- an UML implementation of the conceptual model (see appendix 7.1) which can be reused by digital preservation researchers and developers.
- a machine interpretable implementation of the conceptual model (see appendix 7.2) that can be used by preservation services.
- an example scenario (see chapter 7.3).
- a top-level vocabulary for the entities in the model (see chapter 3). DePICT develops a common top-level structure, and provides guidance to stakeholders on how to use and extend the conceptual model. The top-level vocabulary for each entity can be extended by specialist vocabulary as needed.

- an analysis of the role of digital object properties and characteristics (see section 3.1.4). Interesting relationships between properties of digital preservation objects and their environments occur in the digital preservation process that are not straight-forward to resolve. This thesis investigates how a property ontology can be used to model them explicitly in order to overcome possible misalignments.
- an analysis of constraints that guide digital preservation processes (see section 3.2.3.2.2). In particular, this thesis considers *SignificanceConstraints* one specific form of preservation guiding constraint. It examines the concept of "significance" of the properties of preservation objects in digital preservation, which determines which properties must be preserved over the long-term. It presents a new model that places significance in the hands of stakeholders. The model also extends the domain of *SignificanceConstraints* beyond digital objects to include environments.

  This analysis applies to the digital preservation domain, but may apply to other transformation applications, such as rendering accessible versions of digital objects for disabled users.
- an analysis of how preservation services interact and use preservation metadata dynamically, and of how properties, characteristics and constraints affect the interaction of digital preservation services (see chapter 4).

The model will be validated against real-life standards, tools, policy documents and preservation approaches so that

- the resulting conceptual model of digital preservation is comprehensive, appropriate and readily usable for capturing the main concepts in the domain and for supporting the functional modelling of the domain.
- the analysis of digital object properties, characteristics, constraints, and the interaction of preservation services provides an improved description of the domain.

### *1.2.3 Relevance*

The existence of such a model makes a contribution towards protecting the substantial investments which have been made into the creation of digital assets. It has ramifications in a wide array of sectors:

- memory institutions,
- higher education, and
- industries, which are rich in digital information that needs to be preserved in the longer term.

It provides a conceptual framework

- for scholars who conduct research on digital preservation,
- for preservation experts at institutions who actively preserve their digital collections,
- for digital content owners who specify policies and strategies for their collections,
- for digital preservation tool developers.

Such a model supports implementations of

- digital object repositories,
- preservation metadata dictionaries,
- digital format, technical environment and property registries, and
- digital data management and preservation services.

There is the possibility of significant impact from this model, since it already has started to be integrated into the work of the British Library. The resulting model draws from the PREMIS data dictionary (PREMIS, 2012), but also feeds into it, since the author serves on the Editorial Committee. It will draw from and feed into context and constraints modelling as executed in the TIMBUS project (Dappert, Peyrard, Delve, Chou, 2012).

## 1.3 Research methodology

The main contribution of DePICT is the development of a conceptual model of the digital preservation domain. Conceptual models are used in many areas of computer science, particularly in data, information and knowledge management. Approaches have initially been conceived to support the development of database systems, for example Entity-Relationship modelling techniques (Chen, 1976); object-oriented modelling approaches (Object Management Group, 2011b) were intended to support the development of object-oriented programs, and metadata modelling is used to develop metadata schemata that support the capture of "data about data", as is frequently used in library or online shopping catalogues. But conceptual modelling is applicable to any form of information modelling and should underlie any form of information management.

In order to model a domain one needs a modelling methodology and a notation (or language) in which to capture the model following a consistent set of rules. Notations are often in graphical form as well as in a textual schema notation, which supports the textual serialisation of a model instance for digital processing. A number of the most common approaches are discussed in this section.

From the DePICT conceptual model we can derive metadata definitions in the form of a data dictionary, XML schema or database table; we can derive APIs for interfaces and the service oriented architecture of digital preservation software; etc.

### 1.3.1 Methodology for developing the DePICT model

The DePICT model was developed as original research while the author was working on the Planets project[4] (Farquhar, Hockx-Yu, 2007), the SCAPE project[5] (Edelstein et al., 2011; SCAPE, nd) and the TIMBUS project [6] (Edelstein et al., 2011; TIMBUS, nd). These large scale EU co-funded projects presented an ideal testbed for examining concepts, properties and requirements applied in digital preservation methodologies and tools, and to investigate their information needs and information exchange. The three projects had different foci: interacting preservation services and tools covering the whole of the business-cycle; scalable solutions for large collections or for collections consisting of large, complex or heterogeneous objects; and preservation of processes

[4] Planets, a four-year project co-funded by the European Union 2006 – 2010 to address core digital preservation challenges. www.planets-project.eu/

5 SCAPE, a three-and-a-half-year project co-funded by the European Union 2011 – 2014 to address core digital preservation challenges. www.scape-project.eu/

[6] TIMBUS, a three-year project co-funded by the European Union 2011 – 2014 to address digital preservation of business processes. www.timbusproject.net/

and third-party dependencies with some specialisation on legal issues affecting digital preservation. Being able to study the field under these different perspectives enriched the model and ensured thorough coverage. At the same time the author was employed by the British Library and the Digital Preservation Coalition, which permitted ready access to content owning experts and practitioners who were willing to test the model and to be interviewed about their collections, their decision making approaches and constraints applying to their digital preservation practice. It also permitted access to large-scale digital collections to understand the properties of a large variety of different content-types. It finally also permitted an appreciation for real-life business processes and pragmatic business needs. The author also served on the PREMIS[7] Editorial Committee that strives to provide a data dictionary as *de facto* metadata standard, with which the digital preservation community can capture its digital preservation metadata needs. Intimate familiarity with the dictionary resulted in the author's awareness of short-comings of the current solution, her ability to influence changes to the *de facto* standard, and the ability to closely interact with the user community to understand user needs in practice.

A successful model

- can capture all of the information that needs to be captured to support the functionality required;
- is easily maintained and easily understood by its users by virtue of being slim and tidy, avoiding unnecessary detail and avoiding multiple possible implementations for identical problems;
- encourages interoperability through its clarity of intentions. Different users find it easy to come up with similar implementations for similar problems;
- permits solutions that are natural to the domain and does not require contortions when it is applied;
- is flexible and general enough to accommodate different uses;
- is extensible to increase the level of detail to one that is appropriate to the individual tasks.

The DePICT model's goal is to cover all core digital preservation functions without limiting itself to particular sub-domains or implementation techniques and technologies.

[7] PREMIS, the *de facto* standard on digital preservation metadata. A data dictionary with associated optional XML and RDF implementations. (PREMIS, 2012)

In order to develop a successful conceptual model for a domain, it is necessary to have a comprehensive understanding of it. To gain this understanding for the digital preservation domain, a large number of information sources was analysed as summarised in Table 4. Each information source provided one iteration in the improvement of the model. Each information source was studied in detail. Concepts, properties, and vocabulary were extracted. The concepts were categorized and related, and model requirements were identified. They were then compared to the previous iteration's model, resulting in the addition, removal, combination, refinement or restructure of model elements when gaps became apparent. They were also compared against existing conceptualisations of the domain in order to discover the gaps that currently don't meet user requirements. The results of this gap analysis are reported in chapter 2. Wherever possible the DePICT model was aligned with existing models, if not, DePICT would extend the coverage of existing models. This process was continued until the model reached a stable state where the analysis of new information sources no longer resulted in modifications. Chapter 5 on validation and valuation of the model describes in detail how this methodology was implemented.

Once the stable state was reached, a final, formal conceptual model expressed in UML was created from the collected model requirements. A corresponding appropriate machine-interpretable model as an XML schema was implemented.

The analysis of information sources in the DePICT context allowed for the original interpretation of some particularly interesting issues that had previously been raised in the digital preservation community, but could now be analysed in depth as the entities in question were soundly embedded in a coherent framework. These issues are

- the mismatch between and the relationship of properties that can be extracted from preservation objects to the properties that are used by stakeholders to express their preservation requirements (section 3.1.4.2).
- the role of significant properties (significance constraints) in digital preservation and the relationship between significance constraints and the representation information of the OAIS framework (OAIS, 2002) (section 3.2.3.5).

In a final validation step, the model was used to contribute to the improvement of the PREMIS *de facto* standard. Again, this is discussed in depth in chapter 5 on validation and valuation.

**Table 4: Research methodology approaches**

| Top-down approaches: Model requirements, refinement and validation |
|---|
| • Create a preliminary model from first principles: what scope, context, and functions in digital preservation should be addressed, and what concepts should be present to support them.<br>• Analyse the literature for theoretical descriptions of digital preservation conceptual models.<br>• Analyse the literature for abstract definitions of preservation policies and preservation strategies. |
| **Bottom-up approaches: Model requirements, refinement and validation** |
| • Analyse actual preservation policy and strategy documents drawn from various institution types for their content. They capture many of the concepts that are seen to be important by decision makers.<br>• Interview decision makers to determine factors that influence their preservation decisions.<br>• Compile a list of example constraints found in policy and strategy documents and mentioned in expert interviews.<br>• Study the broad array of preservation services implemented by the Planets project (Farquhar, Hockx-Yu, 2007; Planets, nd). Analyse which information on which concepts is used and produced by them. Perform a gap analysis of which of their aspects are not supported by existing conceptual models.<br>• Study the interaction of the preservation services implemented by the Planets project.<br>• Study the constraints expressed in the use cases collected through the Plato preservation planning tool.<br>• Apply the conceptual model during the design phase of the metadata management component for the British Library's Digital Library System.<br>• Study the functional models for digital preservation in OAIS and Planets.<br>• Learn from existing models, such as PREMIS (2012) and the other work described in the related research chapter 2.<br>• Engage with the PREMIS user community to determine unmet needs.<br>• Develop concrete change proposals to the PREMIS data dictionary to test for practical implementability of DepICT ideas.<br>• Examine how the model fits with the ISO31000 standards for risk management. |
| **Gap Analysis** |
| • Contrast the requirements and the resulting model with existing models, such as PREMIS (2012) and the other work described in the related research section. |

Synthesis

- At each step
    - Extract relevant concepts, properties, relationships and requirements from the information gained;
    - Refine and validate the most current model with the newly found information.
    - Align as much as possible with existing models; extend existing models when necessary.
    - Update the gap analysis to show where existing models do not meet user requirements.
- Create a final, formal conceptual model in UML.
- Design a corresponding appropriate machine-interpretable model (e.g. XML schema).

Valuation

- Prepare in-depth analyses of particularly relevant issues.
- Contribute to the improvement of the PREMIS *de facto* standard.

### *1.3.2 The constructs of a conceptual model for digital preservation*

A conceptual or domain model is used to capture information about a domain, so that defined functions can be performed with the help of this information. As a first step the functions to be supported need to be well understood, scoped and captured and requirements need to be specified, which the model needs to satisfy. The model then specifies the key elements of its domain, which are both the key entities and the correct relationships between them. Furthermore, it specifies the core properties that need to be captured about both entities and relationships to support the desired functionality. The conceptual model defines the domain for these properties, which are the permissible values the properties can take. This often involves the definition of controlled vocabularies, such as lists of permissible values for file format identifiers. Sometimes models include enhanced modelling concepts, such as specialisation (where an entity is a special case of another, from which it inherits properties) or aggregation (where an entity is part of an aggregate entity). Conceptual models should clearly define the meaning of the model elements, eliminate irrelevant information, disambiguate related or similar terms, and provide usage guidance so that the model can be applied in compatible ways by different users and systems. Conceptual models need to be validated against the requirements that were specified for it.

The conceptual model captures the general case of the domain. It can be instantiated in different individual situations. For example, the DePICT model describes the key elements of the domain of digital preservation. It can then be instantiated by a national library that wishes to describe its file format profiling activity which, in turn, lets it understand the composition of its digital collection. But it can equally be used by a computer game manufacturer who wishes to record the significant functional characteristics of a game that needs to be ported to a new platform. The conceptual model should enable its user to clearly and comprehensively capture the information about her instance of the domain.

Conceptual models are design and implementation independent. They specify the elements of the domain without specifying how they will be used in an implementation. They can be a first step in a sequence of modelling approaches. The initial, implementation-independent conceptual model of the domain can be used as the basis for the later development of the design-specific logical model, which specifies the data model to be used (e.g. in a relational database). It can also be used in the implementation-specific physical model that determines the platform-specific implementation details. These design and implementation specifications can be created by hand or through model-driven automatic approaches (Poole, 2001). In model-driven engineering, the standardised elements of the conceptual model instance, expressed in an appropriate domain-specific language (DSL), can be automatically translated into executable representations on various platforms.

### *1.3.3 Modelling languages*

Conceptual models can be designed through a variety of approaches and expressed in a variety of notations. Two popular modelling approaches are object-oriented and entity-relationship modelling. Even though they are intended to support software system or database design, they can be used to develop any type of ontology. Both are supported by formal modelling languages with well-defined semantics and notations that enable the exchange of model information between tools. Ideally they meet the following requirements (Object Management Group, 2011b).

- A formal definition of a common meta-model that specifies the abstract syntax that defines the set of modelling concepts and their properties, as well as the rules for combining these concepts to construct models.
- A detailed explanation of the semantics of each modelling concept. The semantics define, in a technology independent manner, how the concepts are to be realised by computers.

- A specification of the human-readable notation elements.
- Compliance requirements for supporting tools.

#### *1.3.3.1 Object-oriented modelling and UML / OCL*

Object-oriented modelling focuses on

- the set of inter-related, often hierarchical, classes (groups of interacting entities) in the domain,
- their properties , representing the entities' state, and
- the functions ("methods") that are applied to them, representing the entities' behaviour.

The Unified Modeling Language (UML) (Miles, Hamilton, 2006; Bezivin, Muller, 1999; Object Management Group, 2011b) has evolved as the most widely used graphic modelling language to describe these elements of an information system during the design phase of a software system. It is an industry standard created under the auspices of the Object Management Group (OMG) (2011b) that standardises the representation of object oriented analysis and design, and is well supported by a large set of tools. Its notation can, if desired, be translated by a CASE tool into a specific object-oriented programming language, a declarative language or into a database schema.

UML consists of a set of diagram types that capture different aspects of the system at different points of time in the software life cycle, such as Use Case and Activity diagrams for requirements gathering, Class and Object diagrams for model design, and Package and Subsystem diagrams for model deployment. The Class Model is at the core of object-oriented development and design, capturing both the persistent state and the behaviour of the system.

Diagrams in this thesis are based on UML class diagrams. The elements in use in this thesis are classes, depicted as boxes ▭ representing concepts/entities, and the core associations representing relationships between concepts, such as

- association: any relationship between two modelling elements,
- aggregation: a whole-part relationship between an aggregate and its component element,
- composition: a whole-part relationship between a container and its component element with a life-cycle dependency between them,

- generalisation / inheritance: a taxonomic relationship between a general superclass and more specific sub-class element,
- dependency: a change to the independent modelling element will affect the dependent modelling element.

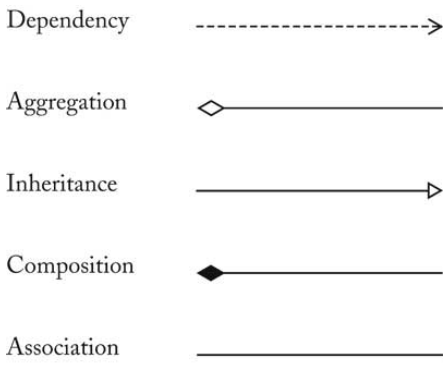



**Figure 4: UML relationship symbols**

Constraints also need to be modelled. The example in Figure 38 is based on the Object Constraint Language (OCL) (Warner, Kleppe, 2003), an addition to UML to capture more complex constraints in an unambiguous way. OCL is a formal language used to specify, in a programming-language-independent way, invariant conditions that must hold for the system being modelled, pre-conditions, post-conditions or queries over objects described in a model. OCL is a pure specification language; therefore, an OCL expression is guaranteed to be without side effects. OCL expressions can be used to specify a state change, for example in a post-condition, but do not affect them.

### *1.3.3.2 Entity-Relationship modelling and ERD*

The entity-relationship modelling (Chen, 1976) approach, introduced by Chen in 1976, represents data and its requirements conceptually, but, unlike object-oriented modelling, does not model their behaviour. It is traditionally a database modelling method associated with relational databases. The corresponding diagrams that capture the models are called entity-relationship diagrams (ERDs) or ER diagrams. ERDs capture entities, relationships, and their attributes (non-relationship properties). An ER entity strictly speaking, is an instance of a given entity-type, which is a class. The term 'entity' is typically casually used to refer to an entity-type / class. In section 1.3.7, this thesis defines the term entity differently, in the object-oriented tradition, and hopefully this will not cause too much confusion, as in the rest of this thesis UML modelling is in use. In the object-oriented approach every object has a unique object identifier which is independent of

property values. In contrast, the ER model uses a minimal set of uniquely identifying attributes as primary keys to identify entities.

Entity-relationship diagrams show entity sets and relationship sets. Cardinality constraints on relationship sets may be captured. Entity sets are represented through rectangles, relationship sets through diamonds. If an entity set participates in a relationship set, they are connected with a line. Attributes are drawn as ovals and are connected with a line to exactly one entity or relationship set.

One can map a class in an object-oriented diagram to one or more entities in an Entity-Relationship Diagram, or vice versa.

### *1.3.4 Implementation-independent textual representations – Data dictionaries*

Text, rather than graphical notations for conceptual models are often desirable. PREMIS (2012) as a metadata model, is described in the form of a textual data dictionary. It is implementation independent and has variously been implemented through XML schemata, data base implementations or RDF implementations.

Figure 11 shows the form in which this information is captured. The properties (called semantic units) are applicable to specific entities and are structured hierarchically. Only semantic units at the leaves of the tree can take values. The permissible values have been taken from a controlled vocabulary that has been captured as SKOS descriptions (Library of Congress, nd-b).

### *1.3.5 Implementation-dependent textual representations - XSD / XML*

Text notations are desirable especially if they are both human and machine-readable and are represented in a non-proprietary format. This means that an instance of a conceptual model can be serialised, i.e. converted into a machine-interpretable format that can be stored or transmitted across a network and understood by many without depending on proprietary software.

A popular choice for exporting a graphical UML model is a textual Extensible Markup Language (XML) schema (Bray et al., 2008). DePICT was implemented as an XML schema, as presented in appendix 7.2. An XML schema defines a set of rules for encoding information in Extensible Markup Language (XML), a format that is both human-readable and machine-readable. Using the XML schema as an implementation guide, an organisation can capture all the information that describes its digital preservation situation in XML. That is to say, the XML schema tells the user

how to model and express the information they need to capture. If for example, the schema specifies

```
<complexType name="EnvironmentType">
    <sequence>
        <element name="environmentName" maxOccurs="unbounded"
                                                minOccurs="0" type="string"/>
        <element name="environmentPurpose" maxOccurs="unbounded"
                                                  minOccurs="0" type="string"/>
        <!-- creation, ingest, preservation, remote access, local access, migration, etc. ->
    </sequence>
</complexType>
```

then the user might use this to describe their situation as

```
    <environmentName> Reading Room Work Station 1</environmentName>
    <environmentPurpose>local access</environmentPurpose>
```

The W3C tutorial (w3schools.com, nd) puts it as follows:

> *"The purpose of an XML Schema is to define the legal building blocks of an XML document, just like a DTD.*
>
> *An XML Schema:*
>
> - *defines elements that can appear in a document*
> - *defines attributes that can appear in a document*
> - *defines which elements are child elements*
> - *defines the order of child elements*
> - *defines the number of child elements*
> - *defines whether an element is empty or can include text*
> - *defines data types for elements and attributes*
> - *defines default and fixed values for elements and attributes"*

### *1.3.6 Modelling the general and specific digital preservation domains*

The diagram in Figure 5 gives an overview of how the general model described in this thesis can be used to create a specific preservation model. The General Model consists of the entities (including their relationships to each other), their properties and vocabulary that are described in DePICT, and the Instantiated Model consists of a specific instantiation to reflect the individual state and constraints of an application.

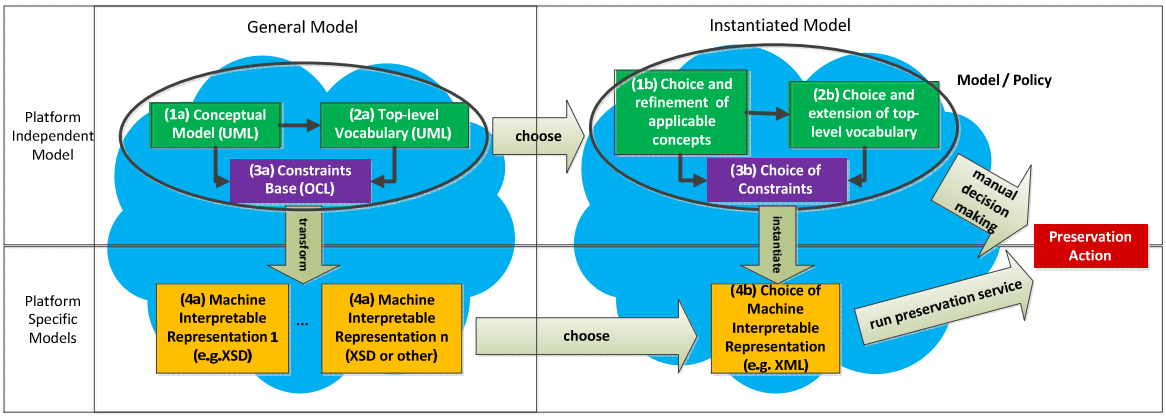



**Figure 5: Modelling institutional constraints**

The numbering in the following text refers to components in the diagram. Numbering including the letter "g" describes components in the general model. Numbering including the letter "i" describes components in an instantiated model.

(1g) The conceptual model, as discussed here, defines the basic entities that are needed in the domain of digital preservation and the relationships between them. They comprise *PreservationObjects*, *Environments*, *Characteristics*, *PreservationActions*, *PreservationRisks*, *Constraints* and others that are introduced in section 3.

(2g) The specific vocabulary defines

- sub-classes of the basic entities,
- properties of the basic entities and their sub-classes,
- allowable values for these properties.

(3g) The constraints base describes sets of constraints which may be contained in digital preservation policies. They are expressed solely in terms of the entities and properties of the conceptual model and its specific vocabulary. They may be parameterised so that they can be instantiated for a specific institution's conditions. An example constraint base is listed in report PP2-D2 (Dappert, Ballaux, Mayr, van Bussel, 2008) derived from an analysis of digital preservation policies and Plato (Becker et al., 2008b) preservation planning requirements trees.

(4g) The entities and relationships in the conceptual model, the specific vocabulary, and the constraints base can be translated into several implementation-specific machine-interpretable representations, for example, based on an XML schema.

(1i) The individual application chooses which of these entities are applicable to its setting and are needed by its preservation service. Since the conceptual model is very concise, in most cases all of the entities would be used.

(2i) The individual application chooses which top-level vocabulary applies to it and extends it with domain specific vocabulary. The individual application also assigns values to the characteristics of its preservation objects and environments if these values are not to be measured automatically, or otherwise specifies the method of obtaining measurements or derivations. It might, for example, need registries of tools, formats, and legislative constraints, and need inventories of its collections, software licenses and staff members.

(3i) The individual application chooses which constraints in the constraints base apply and instantiates them, so that they are now un-parameterised. It specifies additional constraints specific to it. It specifies importance factors, operators, and tolerances.

The outputs of steps (1i), (2i) and (3i) form the core part of a specific preservation model.

(4i) From the choices of steps (1i), (2i), (3i), and the choice of machine-interpretable language, an instantiated machine-interpretable description of the individual application's constraints is derived. This serves as a basis for automated preservation services. Many constraints in preservation guiding documents, such as policies, especially on higher institutional levels, may not be machine-interpretable, but it can still be useful to represent a machine-interpretable subset for automatic evaluation.

Preservation services can manually evaluate the constraints and the state described in the platform-independent, instantiated model (1, 2, 3 i), or automatically evaluate the constraints and the state described in the machine-interpretable model (4i). They can then identify which preservation actions can best satisfy the constraints under the given state.

### *1.3.7 Conceptual Modelling in DePICT*

Chapter 3 of this thesis establishes the requirements that must be met by a conceptual model of the digital preservation domain. It identifies the main entities, their relationships and properties and describes how they are used in practice. In appendix 7.1 this informal description is translated into a formal UML model. And in appendix 7.2 this UML model is translated into an implementation specific XML implementation. The basic vocabulary for talking about entities, properties, values, and so on is taken from object-oriented modelling as defined in the OKBC

protocol in (Chaudhri, Farquhar, Fikes, Karp, Rice, 1998). The core terms in this vocabulary that are relevant to DePICT are:

- Entity – Anything whatsoever.
- Class – A class is a set of entities. Each of the entities in a class is said to be an instance of the class.
- Property – A property is an entity that is not class that names a relationship.
- Characteristic – A property / value pair associated with an entity. The value is an entity. This relationship is illustrated in Figure 6.
- Constraint – A Boolean condition involving expressions on entities.

A sub-class relationship is a property that names a relationship between two classes, a sub-class and superclass, in which the sub-class inherits properties of the superclass.

Unless otherwise specified, a characteristic is directly associated with an entity. Furthermore, a property applies to a class if it can be meaningfully associated with some instances of the class. Under this terminology, it is clear that a characteristic (property / value pair) may be preserved by a preservation action, but that the abstract property cannot be.

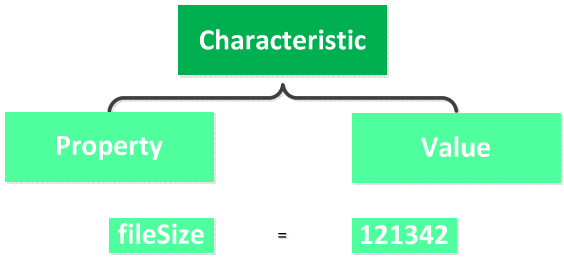



**Figure 6: Properties and Characteristics**

One can use this language in the domain of digital objects and preservation.

For example,

- `File` is a class;
- `f1.txt` is an instance of the class `File`;
- `fileSize` is a property;
- the property `fileSize` applies to `File`;
- File `f1.txt` has the characteristic `fileSize` = 131342;

- *Collection* is a sub-class of *IntellectualEntity;*
- *MyDigitalCollection* is an instance of the class *Collection;*
- *MyDigitalCollection* has characteristics *numberOfObjectsInTheCollection* = 850, *valueOfTheCollection* = 2000.

Important additional information about a property or characteristic, such as how a value can be encoded for a property or is encoded for a characteristic, applicable units of a property or the actual unit of a characteristic, or the algorithm or tool that can be used to compute the value for a property or have actually been used for a characteristic can be specified using facets. Facets that are particularly important for digital preservation are specified in the conceptual model below.

## 1.4 Digital preservation research in EU projects

Under the 6th (2002-2006) and 7th Framework Programme (2006 – 2013), the European Commission has funded digital preservation research in 15 projects (CASPAR, Planets, DPE – Digital Preservation Europe, WePreserve, APARSEN, ARCOMEM, BlogForever, ENSURE, KEEP, PrestoPRIME, PROTAGE, SCAPE, SHAMAN, TIMBUS, Wf4Ever) (CORDIS, nd). These projects have investigated many different aspects of digital preservation. The DePICT model was validated against the concepts used in the projects described here.

### 1.4.1 Planets

Planets (Farquhar, Hockx-Yu, 2007; Planets, nd) was a large-scale EU co-founded digital preservation project that was led by the problem owners, but also included technology providers and researchers. It moved past research to the practice of digital preservation, took significant steps towards building a practice oriented digital preservation community, and succeeded in understanding and testing its products against real-life problems. It developed tools and methodologies to enable large-scale content holding organisations to take care of their digital collections.

Planets products support the following functionalities for the organisations:

“

- Express preservation policies
- Profile digital collections
- Identify and diagnose problems in digital collections
- Compare different treatment plans
- Select and implement treatments
- Verify that the treatment was successful
- Know which solutions work through empirical evidence
- Encourage vendors and service providers to provide these capabilities ”

(Farquhar, 2007)

The Open Planets Foundation (OPF, nd) was created as an arena where research outputs, tools and services developed within the Planets project could be sustained, enhanced and developed into an Open Source digital preservation environment. It is an international practitioner and

developer community with a focus on practical solutions. As well as supporting training, advice, technology development and community building, its declared goal is to develop services that enable its members to

- "Make informed and accountable preservation decisions, based on the best available evidence.
- Assess the preservation needs of your organisation, collections and users.
- Build, evaluate and execute plans to address any problem areas.
- Analyse and verify the results.
- Help to create the evidence base about preservation tools and their behaviour.
- Take advantage of tools and services for your preservation planning, characterisation, conversion, migration, emulation and database archiving activities.
- Access stable hosted digital preservation services on-line.
- Test and evaluate digital preservation approaches prior to implementation without the need to configure hardware or install any software."

(OPF, nd)

**Relationship to DePICT**

Planets organised the digital preservation problem into a "characterisation, planning, action" cycle, which is used in this thesis in chapter 4. It supported a broad set of methodologies, including migration, emulation, complex migration through combining multiple emulators, and database preservation. This broad approach provided a good basis for validating a methodology-agnostic conceptual model. The core of the work in this thesis was developed as theoretical underpinning for the Planets tools and services and was co-funded under the Planets project.

### *1.4.2 SCAPE*

The SCAPE project (SCAlable Preservation Environments) (Edelstein, et al., 2011; SCAPE, nd) is a follow-on project to Planets with the specific goals of developing scalable services for planning and execution of institutional preservation strategies on an open source platform that orchestrates semi-automated workflows for large-scale, heterogeneous collections of complex digital objects. It addresses known limitations of the functional components of a digital preservation system. With respect to scalability, it improves tools to handle large numbers of

digital objects, physically large digital objects, complex digital objects (nested and linked), and heterogeneity of digital objects. It also aims to enhance the coverage and quality assurance of the functional components (Characterisation, Action Services and Quality Assurance Components) that had been developed in Planets. It will also develop new tools where necessary, apply proven approaches like image and patterns analysis in novel ways, integrate with policy-driven preservation watch and planning, ensure interoperability between services, and deploy its outputs on the distributed, parallel SCAPE platform.

**Relationship to DePICT**

The SCAPE approach was useful for validating the applicability of DePICT to a digital preservation scope that includes highly-scalable objects and scenarios.

### *1.4.3 KEEP*

The KEEP project (Keeping Emulation Environments Portable) (KEEP, nd) investigated the use of emulation as a digital preservation methodology. It developed emulation services that support the rendering of obsolete static and dynamic digital objects through tools that reproduce their original creation environment or that enable them to be migrated to another environment. The highly portable KEEP Emulation Framework hosts them, as well as third party emulators and automatically determines the correct execution environment by analysing the files to be supported. The KEEP Virtual Machine prototype constructs independent execution environments on top of native software and hardware platforms. By developing transfer tools, KEEP enabled access to digital content stored on outdated data carrier technology, such as floppy discs, as a basis for the emulation work. The project also investigated legal issues that affect emulation-based systems.

**Relationship to DePICT**

Of particular interest to the conceptualisation of the digital preservation domain was KEEP's effort to define a metadata model for capturing entities crucial to emulation. The TOTEM Database Tool (Delve, 2011; Delve, Anderson, 2012; TOTEM, nd) provides access to such information. TOTEM is described in section 2.2.5.2.3.

### *1.4.4 TIMBUS*

The EU co-funded TIMBUS project (Edelstein, Factor, King, Risse, Salant, Taylor, 2011; TIMBUS, nd) addresses the challenge of digital preservation of business processes and services to ensure their

long-term continued access. TIMBUS analyses and recommends which aspects of a business process should be preserved and how to preserve them. It delivers methodologies and tools to capture and formalise business processes on both technical and organisational levels. This includes their underlying software and hardware infrastructures and dependencies on third-party services and information. TIMBUS aligns digital preservation with well-established methods for Enterprise Risk Management (ERM), feasibility and cost-benefit analysis, and business continuity management (BCM). It evaluates them in three use cases: Engineering services and systems for digital preservation, civil engineering infrastructures and eScience. It is conducted by a consortium of industry, research and SME partners from across Europe.

**Relationship to DePICT**

The TIMBUS project is of particular value for the DePICT conceptualisation as it stretches the boundaries of the definition of *Preservation Objects* and computing *Environments* to include business process related metadata and third-party services.

# 2 Existing conceptualisations of digital preservation

This chapter plays a dual role. Firstly, it gives an overview over related work on conceptualising the digital preservation domain. Secondly, in each section, the related work introduced is related to the DePICT model. This also illustrates gaps in the existing work that do not satisfy the users' requirements that were derived in this research. Since the modelling requirements are not introduced until chapters 3 and 4, this means that, for a thorough understanding of this gap analysis, it will be necessary to read the subsequent chapters and to come back to the gap analysis later. They will, however, provide an intuitive idea about the motivations that led to the development of DePICT.

In order to understand the activities that have taken place in the area of conceptual modelling of digital preservation, this section investigates work done in four areas:

- A framework for archival information systems;
- Digital preservation metadata
    - for general-purpose descriptions ,
    - for describing specific aspects of the digital preservation domain,
    - for creating registries that support digital preservation;
- Constraints on preservation objects, environments and preservation actions;
- Functional models of digital preservation.

## 2.1 A framework for open archival information systems - OAIS

One of the anchors of conceptual modelling in digital preservation is found in the OAIS model. In 2002, the Reference Model for an Open Archival Information System (OAIS) (CCSDS, 2002) provided a framework to unify the concepts and terminology in the digital preservation community. This ISO 14721:2003 Reference Model, defined by recommendation CCSDS 650.0-B-1 of the Consultative Committee for Space Data Systems, is an architectural, information, and functional framework for archival systems dedicated to long-term digital preservation. Its functional entities are depicted in the well-known diagram in Figure 7.

The 3 types of stakeholders for an OAIS repository, either human or automated processes, are Producers, who submit digital information objects to the OAIS, Consumers, who obtain digital information objects from the OAIS, and Management, who is responsible for managing the OAIS.

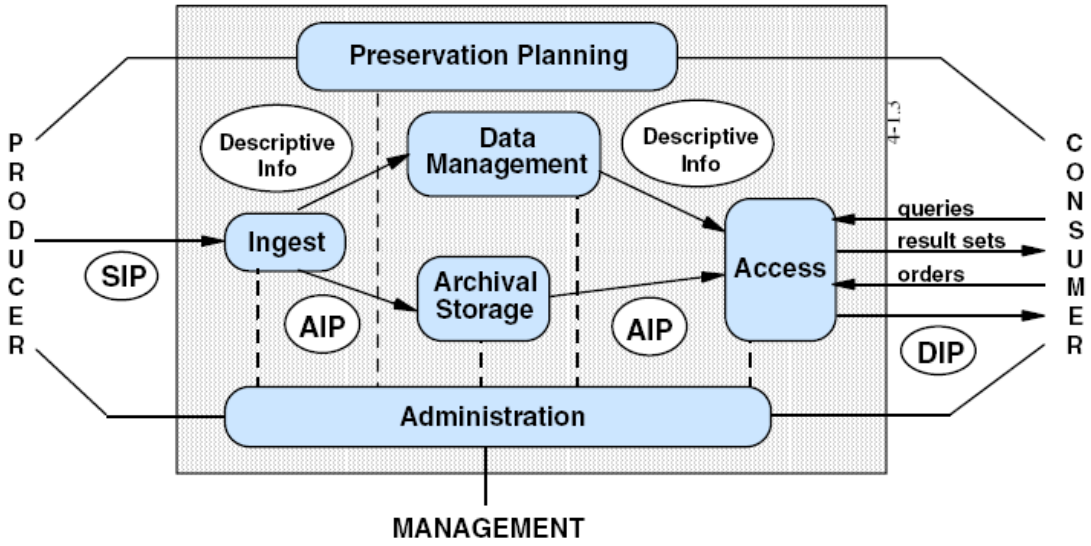



**Figure 7: OAIS functional entities (Figure 4.1 in CCSDS, 2002)**

The 3 types of Information Package (IP) in the OAIS information model are the Submission Information Package (SIP), the Archival Information Package (AIP) and the Dissemination Information Package (DIP). SIPs are submitted to the OAIS by the Producer and can undergo various stages of transformation before they are ingested into the OAIS. AIPs reflect the form and content in which the digital information object together with its metadata is stored in the OAIS. The 2012 version of the OAIS (CCSDS, 2012) introduces AIP versions, which result from a digital preservation action on a previous AIP, and AIP editions, which reflect functionally improved content of previous AIPs. DIPs are the form and content in which they are distributed to a Consumer. Since, for one AIP, there may be various Consumers representing different Designated Communities, there may be multiple DIPs. A Designated Community describes a set of Consumers with its own set of requirements for the preservation of the digital information object and shared knowledge about how to interpret it, which distinguish it from other Designated Communities. A fourth category of information is Descriptive Information, which is search or discovery metadata that helps in finding the object in the repository.

The OAIS is associated with a detailed information model for an AIP that was further developed by the Online Computer Library Center and the Research Libraries Group (OCLC/RLG), 2002), as depicted in Figure 8. Key concepts are those of Content Data Object and Representation Information. The Content Data Object is the primary set of bit-sequences that is necessary to render the digital information object that must be preserved. It does not use the term 'data' in the scientific sense, but comprises documents, images, software and other digital objects that must be preserved. Representation Information is "the information that maps a Data Object into more meaningful concepts" (CCSDS, 2002). Examples for a specific *.docx* file would be its file

format specification that defines how to interpret the bit sequences, a list of software tools that can render it, hardware requirements, the language in which the contained text is written, and contextual information that states the author, purpose and time of its writing. This is needed because digital data objects are normally not self-descriptive and require very specific intermediary tools for access by humans and specific knowledge for interpreting them that may not commonly be available amongst its Designated Community. "Data interpreted using its Representation Information yields Information", which is called Content Information (Object). An Information Package contains Data Objects, their Representation Information and additional metadata, called Preservation Description Information (PDI), that support the management of the Information Object. The PDI consists of reference information for identification purposes, context information that describes the relationship to other objects, provenance information regarding document ownership, stewardship and historical changes to the information object, fixity information to ensure its authenticity, and Access Rights information (as of OAIS, 2012).

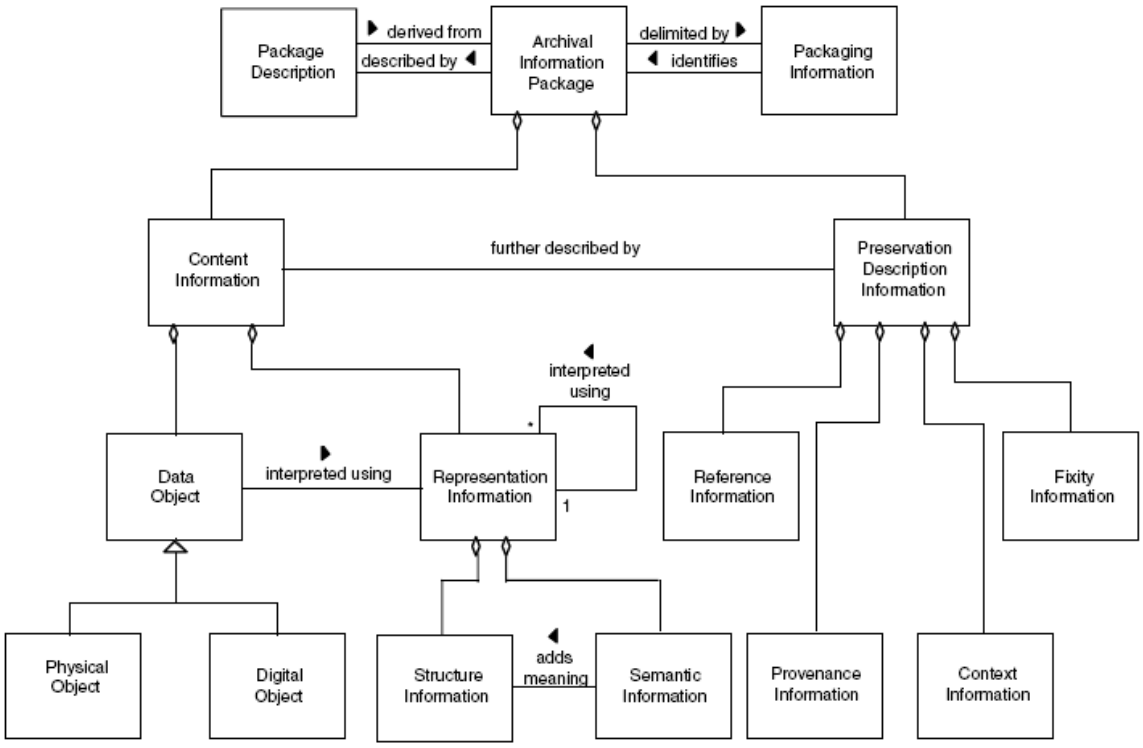



**Figure 8: Archival Information Package (Detailed View) (Figure 4.18 in CCSDS 2002)**

The 7 types of functional preservation services in the OAIS are Ingest, Archival Storage, Data Management, Administration, Preservation Planning, Access and Common Services supplied by any IT system.

The OAIS framework is now in its third revision (CCSDA, 2002; CCSDS, 2009; CCSDS, 2012).

**Relationship to DePICT**

The purpose of the OAIS model is to provide a framework for the implementation of an archival information system and a benchmark against which to test it. The OAIS framework does not cover the whole lifecycle of a digital object, but just its state while it is held in an OAIS repository. This is a very important phase of a digital information object's lifecycle and the OAIS model has contributed greatly towards unifying terminology and approaches in digital preservation with regard to repositories. Practically, however, digital information objects often spend a large part of their lifecycle outside an OAIS repository and many preservation services do not act within the OAIS repository. Because of this the model does not capture all conceptual aspects of digital preservation. For example, OAIS does not have a concept of expressing *Risks* or *Constraints* which guide digital preservation processes. While there is a notion of significant *Properties*, they are, as in PREMIS (2012), restricted to one *PreservationObject* and do not specify characteristics of future *PreservationObjects* derived from current *PreservationObjects*. OAIS does not explicitly model the relationships between subsequent *Representations* of an *IntellectualEntity* (or, for that matter, between any Information Objects or their parts). It does not model *Constraints* that would guide the process of deriving them.[8] OAIS also does not distinguish between physical and logical objects. OAIS Data Objects are probably closest to the *Bitstreams* and *Representations* of DePICT. Data Objects together with their Representation Information create an Information Object. The latter is, however not the same thing as an *IntellectualEntity* in DePICT. *IntellectualEntities* can have Representation Information of their own (e.g. their *Environments*) and need not have direct Data Objects associated with them (such as a journal title). OAIS has no notion of different *Representations* of the same Information Object other than the ability to model Information Objects' context. It also does not always model to the level of detail that is needed to express many of the modelling requirements specified in the body of this thesis.

On the other hand, OAIS makes distinctions that are not explicitly modelled in DePICT. DePICT introduces an *Agent* entity that can be instantiated as needed by the user. One can choose to use the OAIS Producer- Consumer - Management classification as an *Agent*'s refinement when working within DePICT. Similarly, DePICT introduces a *PreservationService* entity that can be further categorised by the OAIS functions. OAIS details the composition of an Information Package as Data Object, Representation Information and Preservation Description Information. Again, DePICT categorises *PreservationObjects* along a more generic intellectual – logical – physical

[8] However, OAIS 2012 is just now introducing Transformational Information Properties that let you specify which properties need to be kept after a transformation.

dimension, but does not prescribe which types of information need to be captured. This flexibility in DePICT is intentional, in order to ensure its applicability to different digital preservation use cases. The individual use case, such as the archival repository functions of the OAIS, can then populate the model with the appropriate categories for their context.

## 2.2 Digital preservation metadata

Metadata is "data about data" (Duval, 2001). It is "structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use or manage an information resource" (NISO, 2004). Digital preservation metadata are the information that is essential to ensure long-term accessibility of digital resources. They need to be preserved together with the digital data objects in order to enable those responsible for their care to achieve their long-term preservation goals as defined in section 1.1.2. It is very hard, and sometimes impossible, to extract, locate or recreate metadata that describes essential features of the digital object and its computing environment after it has fallen into obsolescence. An important task of the digital preservation community is to ascertain which metadata is needed for long-term access.

A metadata model is one form of a conceptual model. This section reviews digital preservation metadata approaches as they pertain to metadata content. The different ways of formalising metadata are discussed in section 1.3.

### 2.2.1 Digital preservation metadata evolution

Analyses of the goals of long-term digital preservation have led to a good understanding of the types of metadata that are needed, at the least for safe-keeping of `PreservationObjects` in an archival information system. Good overviews of digital preservation metadata issues are provided in Caplan (2006), Lavoie and Gartner (2005) and Dappert and Enders (2010). In 2002, the Reference Model for an Open Archival Information System (OAIS) (CCSDS, 2002) provided a framework to unify the concepts and terminology in the community as described in the previous section. The associated information model developed by the OCLC/RLG Working Group on Preservation Metadata (2002) defines categories for preservation metadata. It does, however, not define which specific metadata should be collected or how it should be implemented in order to support preservation goals.

In the early days of digital preservation, there were several uncoordinated efforts to define institution-specific sets of metadata elements (e.g. CURL Exemplars in Digital Archives Project (CEDARS Project, 2002); NEDLIB (Lupovici, Masanès, 2000); National Library of Australia (PADI, nd); National Library of New Zealand (Searle, Thompson, 2003)). These efforts were soon merged into a smaller number of coordinated international activities that aimed to define sharable preservation metadata specifications. This would ensure interoperability—the ability to exchange amongst institutions and to understand the digital object metadata and its digital content. In 2005 the Preservation Metadata Implementation Strategies (PREMIS) data dictionary (PREMIS, 2005),

that consolidated earlier efforts, produced a conceptual model and concrete metadata dictionary for implementers of digital preservation services. Now in version 2.2 (PREMIS, 2012), it has been widely accepted and plays a key role in creating coherence in the digital preservation metadata community. PREMIS provides a foundation to support interoperability across systems and organisations.

Many of the entries in today's data dictionaries are, however, still vague. They await increased practical experience to establish the proper level of granularity. Coming out of the OAIS tradition, they also tend to be focused on statically recording characteristics and events rather than on dynamically supporting preservation processes. Currently, few metadata specifications contributing to digital assets' long-term preservation are sanctioned by national or international standards bodies. Some, like PREMIS (PREMIS, 2012) or METS (Metadata Encoding and Transmission Standard) (METS, nd), have the status of *de facto* standards with well-defined community processes for maintaining and updating them. While communities have a strong desire for long-lasting stable metadata standards, they continue to evolve as the number of repository implementations and applications grows. Experience remains too limited to set a preservation metadata standard in stone.

**Relationship to DePICT**

This thesis evaluates its conceptual model against the PREMIS data dictionary *de facto* standard. Results coming out of the DePICT work have already been fed into the PREMIS Editorial Committee work and will result in an altered data model in version 3 of the PREMIS data dictionary.

### *2.2.2 Digital preservation metadata categories*

The specific metadata needed for long-term preservation falls into four categories based on basic preservation functional groupings:

- Descriptive metadata

  describes the intellectual entity through properties, such as author and title, and supports discovery and delivery of digital content. It may also provide an historic context, by, for example, specifying which print-based material was the original source for a digital derivative (source provenance).

- Structural metadata

  captures physical structural relationships, such as which image is embedded within which website, as well as logical structural relationships, such as which page follows which in a digitized book.

- Technical metadata for physical files

  includes technical information that applies to any file type, such as information about the software and hardware on which the digital object can be rendered or executed, or checksums and digital signatures to ensure fixity and authenticity. It also includes content-type specific technical information, such as image width for an image or elapsed time for an audio file.

- Administrative metadata

  includes provenance information of who has cared for the digital object and what preservation actions have been performed on it, and rights and permission information that specifies, for example, access to the digital object, including which preservation actions are permissible.

Other analyses and frameworks use somewhat different categories of preservation metadata. No matter which categories are used, however, they are never clear-cut or unambiguous. A semantic unit can support several preservation functions and, therefore, fall into several categories. For example, the semantic unit *fileSize* can support both search (e.g., by letting a user search for small images only) and technical repository processes which depend on file size.

The term semantic unit is borrowed here from the PREMIS data dictionary. Semantic units are the properties that describe the digital objects and their contexts or relationships between them. The term metadata element, in contrast, is used to specify how to implement that semantic unit in a given metadata implementation specification.

**Relationship to DePICT**

The process of studying digital preservation metadata dictionaries, ontologies or frameworks reveals what metadata their creators have deemed important in the domain space. This means that they identify at least the subset of the metadata within the conceptual model that are useful for preserving, together with digital data objects. Unfortunately they do not identify the entities that are used dynamically during executing preservation services. DePICT focuses on end-to-end

digital preservation solutions that test the flow of preservation metadata across multiple digital preservation services.

Digital preservation metadata frameworks are also always built on a data model of the digital preservation domain that is worth analysing. They define the entities that need to be described by semantic units. Traditionally they are the digital objects themselves, both as abstract, intellectual entities and as physical realisations in the form of renderable or executable file sets. Entities can also describe a digital object's hardware, software, and societal environments, rights and permissions attached to them, software and human agents involved in the preservation process, and events that took place during the digital object's life cycle. This thesis examines the issue of which entities should be described, how they should be described and how they relate to each other if the desired digital preservation functionality is to be supported.

### *2.2.3 Combining digital preservation metadata specifications*

Because of the breadth of metadata needed to support the full range of digital preservation goals and the variety of scenarios in which digital preservation is applied, it does not make sense to create one monolithic data dictionary to be used by all and to apply to all situations. Many years of expertise and effort have already gone into specifying metadata dictionaries or implementation specifications for subsets of the four categories listed above that are also used to support functions outside digital preservation. There is no point in trying to reproduce or outdo these efforts. Additionally, it is not possible to define one set of metadata that applies equally to all content types or organisation types. Archival records, manuscripts, and library records, for example, require different descriptive metadata; images, text-based documents, and software source code require different technical metadata. Because of this, a number of metadata definition efforts have evolved, both in a content type- or organisation type-specific space and a preservation function space. Different metadata specifications can be combined by using a container metadata schema, such as METS (METS, nd) that defines metadata categories, and relationship and identifier mechanisms through which descriptions in different specifications can link to each other. Figure 9 illustrates this in a very simplified way. Several of these initiatives have reached the status of a standard or are *de facto* standards.

In order to be flexible and apply to a wide range of contexts, general preservation metadata and metadata container specifications try to avoid content and organisation specific semantics. For example, general digital preservation metadata models may capture the `fileSize` of files, since there are no digital representations of content that do not involve at least one file, even if the exact way of determining the file size may depend on an operating system. It would not, however,

capture the *issueNumber*, which applies to serials but not books, or the *resolution*, which applies to images but not text. On the other hand, if the preservation metadata model is to equally apply to digital and non-digital elements, the property *fileSize* must not be a mandatory part of the model.

**Relationship to DePICT**

DePICT is a high-level conceptual model that applies to all digital preservation scenarios. It defines entities that apply universally. These entities can be made more specific by creating sub-classes and by refining the entity space hierarchically. DePICT illustrates how it can be extended through common refinement categories but does not define them as part of the conceptual model.

<table>
<tr><th>Function Types</th><th colspan="4">Content and Organisation Type-Specific Variants</th></tr>
<tr><td>Metadata Containers<br><br>Two examples of container specifications are METS (METS, nd) and MPEG-21 DIDL (ISO/IEC, (2005).</td><td colspan="4">  <br><br>Content and Organisation Agnostic Metadata</td></tr>
<tr><td>General Preservation Metadata<br><br>Two examples of preservation specific metadata specifications are PREMIS (PREMIS, 2012) and LMER (Deutsche National Bibliothek, nd)</td><td colspan="4">  <br><br>Content and Organisation Agnostic Metadata</td></tr>
<tr><td>Descriptive Metadata<br><br>This includes both general purpose approaches, such as Dublin Core[9] (DC, nd), and library or archive community approaches, such as MODS[10] (Mods, nd), MARC[11] (MARC, nd), EAD[12] (EAD, 2002).</td><td>  <br>Manu-scripts</td><td>  <br>Archival Records</td><td>  <br>Books</td><td>  <br>…</td></tr>
<tr><td>Content-Type Specific Technical Metadata<br><br>Two examples of content type-specific metadata are the ANSI/NISO Z39.87 standard (ANSI/NISO, 2006) for Digital Still Images, and the textMD (textMD, nd) specification. For text metadata.</td><td>  <br>Images</td><td>  <br>Audio-video</td><td>  <br>Text</td><td>  <br>…</td></tr>
</table>

**Figure 9: The space of digital preservation metadata efforts**

[9] The Dublin Core (DC) is a metadata element set consisting of a vocabulary of fifteen properties for the purpose of resource description and discovery of a broad range of resources.

[10] Metadata Object Description Schema (MODS) is an XML schema for a bibliographic metadata element set for a variety of purposes, particularly for library applications.

[11] MAchine-Readable Cataloging (MARC) is an international bibliographic standard developed by the Library of Congress in the 1960s. MARC 21 is the most used version.

[12] Encoded Archival Description (EAD) is a metadata element set for archives.

### 2.2.4 General digital preservation metadata

#### 2.2.4.1 Core digital preservation metadata PREMIS

PREMIS (PREservation Metadata: Implementation Strategies) (PREMIS, 2005; PREMIS, 2008; PREMIS, 2011; PREMIS 2012; PREMIS, nd) is one attempt at specifying the metadata (called semantic units in PREMIS) that is needed to support core preservation functions; in fact, it is the current *de facto* standard for doing so. Core preservation metadata is relevant to a wide range of digital preservation systems and contexts, and it is what "most working preservation repositories are likely to need to know" to preserve digital material over the long-term. This includes administrative metadata, but also generic technical metadata that is shared by all content types. It permits the specification of structural relationships between entities if this is relevant for preservation functions, but users may choose to instead use the structural relationships offered by a container metadata specification.

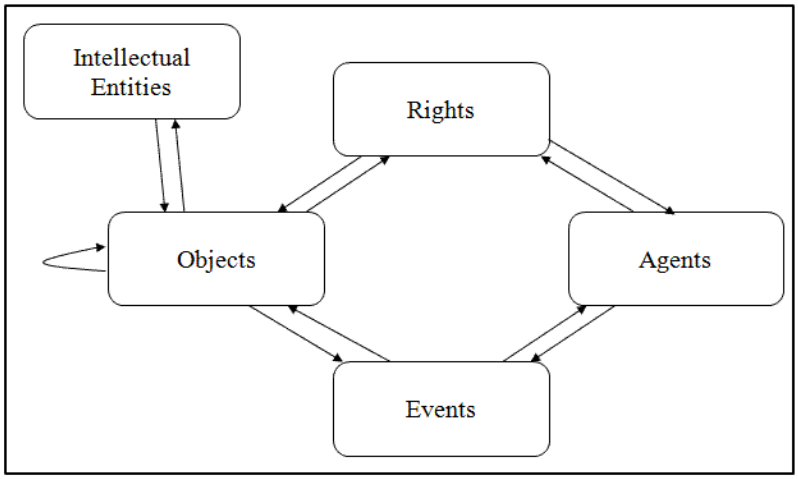



**Figure 10: The PREMIS data model (PREMIS, 2011)**

PREMIS defines a common data model (illustrated in Figure 10) to encourage a shared way of thinking about and for organising preservation metadata. It has Object, Event, Agent, Right and IntellectualEntity as entities. The data dictionary permits relationships between the entities as are indicated through the arrows in Figure 10.

The semantic units that describe the entities in this data model are rigorously defined in PREMIS's data dictionary. PREMIS supports specific implementations through guidelines for their management and use and puts an emphasis on enabling automated workflows. It makes, however, no assumptions about specific technology, architecture, content type, or preservation strategies. As a result, it is "technically neutral" and supports a wide range of implementation architectures. For example, metadata could be stored locally or in an external registry (such as a shared file format registry); it could be stored explicitly or known implicitly (e.g., all content in the

repository are newspaper articles). It does not even specify whether a semantic unit has to be implemented through a single field or through more complex data structures. Nonetheless, the PREMIS Editorial Committee maintains optional XML and RDF schemas for the convenience of the community. While PREMIS is very flexible about possible repository-internal implementations, in order to improve interoperability, it is more restrictive on cross-repository information package exchange. An example PREMIS data dictionary entry for the semantic unit size is depicted in Figure 11.

Given the wide range of institutional contexts, PREMIS cannot be an out-of-the box solution. Users have to decide how to model their specific application, what business functions need to be supported, which semantic units need to be captured to support them, and how to implement them. In addition, they need to decide on all metadata that is necessary to manage the content that is not captured in the core preservation metadata.

| **Semantic unit** | **1.5.3. size** | | |
|---|---|---|---|
| **Semantic components** | **None** | | |
| **Definition** | The size in bytes of the file or bitstream stored in the repository. | | |
| **Rationale** | Size is useful for ensuring the correct number of bytes from storage has been retrieved and that an application has enough room to move or process files. It might also be used when billing for storage. | | |
| **Data constraint** | Integer | | |
| **Object category** | **Representation** | **File** | **Bitstream** |
| **Applicability** | Not applicable | Applicable | Applicable |
| **Examples** | | 2038937 | |
| **Repeatability** | | Not repeatable | Not repeatable |
| **Obligation** | | Optional | Optional |
| **Creation / Maintenance notes** | Automatically obtained by the repository. | | |
| **Usage notes** | Defining this semantic unit as size in bytes makes it unnecessary to record a unit of measurement. However, for the purpose of data exchange the unit of measurement should be stated or understood by both partners. | | |

**Figure 11: Example PREMIS Semantic Unit (PREMIS, 2011)**

**Relationship to DePICT**

As in DePICT, the PREMIS data model has Representations, Files and Bitstreams as sub-classes of Objects (DePICT:*PreservationObjects*). The PREMIS Bitstream is restricted to a bit-stream within one file. DePICT *Bitstreams* are kept general and can consist of sets of bit-streams which can span several files because the bits representing *Characteristics* of *IntellectualEntities* may not necessarily align with byte boundaries (e.g. when they are extracted from a compressed file directly or if *Characteristics* are represented as bitmaps). They may span several files (e.g. large files may be split with a Unix "split" command, data may be streamed into containers of a fixed file-size, such as ARC, data may be split over several files to optimise access).

DePICT distinguishes between the logical file (*RepresentationBitstream*) and the actual file (*Bitstream*). This enables users to model the logical file, with *Characteristics* such as the file's ideal checksum, in contrast to the individual realisations of this file, which might have, for example a *Characteristic* "actual checksum" which can vary from the checksum of the logical file if there is a file corruption. There may be several actual files for one logical file, if there are multiple copies held. Actual files have a location, logical files do not.

The PREMIS data model does not consider IntellectualEntities a sub-class of Objects. As a consequence, PREMIS depends on container metadata schemas for capturing, for example, the IntellectualEntity's descriptive metadata; the data dictionary is not self-contained. It also means that the data model is not as compact and uniform as it could be, and it cannot directly specify Events or RightsStatements, or attach Agent information to IntellectualEntities. IntellectualEntities in PREMIS are not yet fleshed out and the PREMIS Editorial Committee is currently considering how this can be improved for version 3.0. based on the work presented in DePICT.

Significant Properties in the PREMIS model exist but are not as fully defined, as *SignificanceConstraints* are in the DePICT model. There are, for example, no tolerance or importance factors. They can only specify individual *Characteristics* that must be preserved, rather than expressing *Constraints*, which might, for example, specify allowable modifications or post-conditions. They can only be attached to one Object at a time rather than to *Environments* or combinations of *Environments* and *PreservationObjects*. It is important to be able to specify for a business rule (or *Constraint*) under what context it applies. If one stores it with Object, which is currently the case in PREMIS, then that is the only and implicit

context. *SignificanceConstraints*, and in fact *Constraints*, should be a primary entity in the data model rather than subordinate to Object.

How Environments are dealt with in PREMIS is discussed in depth in section 2.2.5.2.1. DePICT has greatly contributed towards improving this specification by being incorporated into the work of the PREMIS Environment Working Group in 2012 (Dappert, Peyrard, Delve, Chou, 2012).

While specific *Properties* are modelled in some depth within PREMIS, PREMIS does not have a generic, rich specification of *Properties* that takes account of *ValueOrigins* and does not offer a meta-level on which to describe the properties of *Properties* and their relationships to other *Properties*. This is not in scope for PREMIS.

*PreservationActions* and *PreservationRisks* are outside the scope of the PREMIS data model.

The Event, Agent and Right entities of PREMIS are adopted for DePICT. They are not modelled in detail in this thesis. Several DePICT entities form digital preservation specific sub-classes to them. It is becoming apparent, however, that there is a need for a richer Event model in PREMIS. For example, if you have an n:m migration, e.g. creating one pdf from multiple files, or creating multiple spreadsheets from one database file, it is very cumbersome and verbose in PREMIS at the moment. For succinctness' sake, PREMIS does not always implement entities that actually are Events and Agents explicitly as such. For example, information about a creating application is modelled as *Properties* of an Object rather than modelled as an Agent and the information about the creating event is, similarly, modelled as a *Property* of an Object rather than as an Event. This provides a convenient shortcut and leads to less verbose XML implementations, but sacrifices the cleanness of the model and makes it harder to adapt the standard to new use cases or implementations. In the DePICT XML implementation such shortcuts are mostly avoided to maintain clean modelling principles. But that is not to claim that, in practical implementations, they will not have to be sacrificed occasionally.

Most of the differences listed in this section are due to the fact that PREMIS was conceived as a data dictionary to capture preservation metadata for digital information objects in OAIS (CCSDS, 2012) repositories. It was not conceived to support dynamic digital preservation actions or end-to-end life-cycles. Figure 12 summarises coarsely how the DePICT and PREMIS entities relate. Entities depicted in blue are concepts that are largely shared, entities depicted in violet are shared concepts that have different extent or take a different role in the model and entities depicted in pink are concepts that are covered in DePICT only.

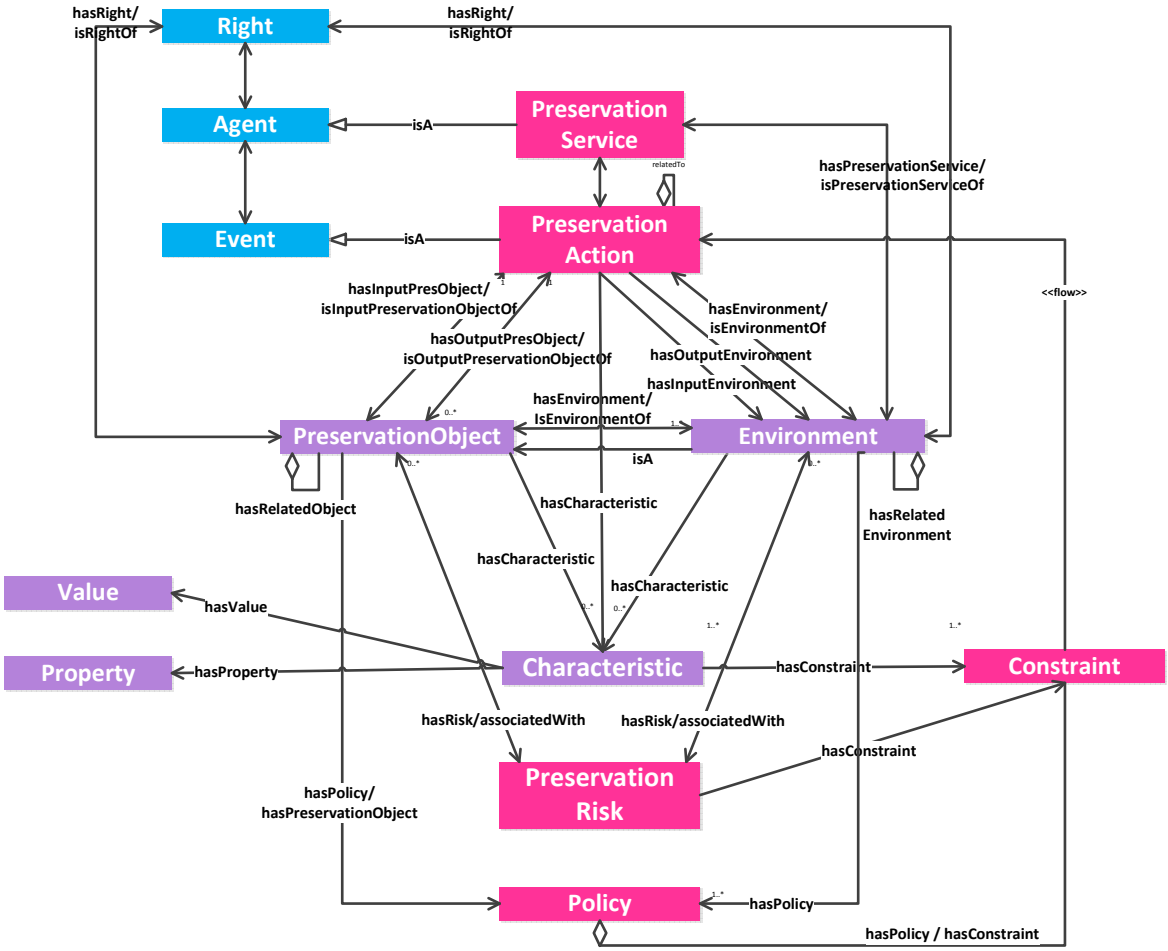



**Figure 12: DePICT in relationship to the PREMIS model.**
**Blue: largely shared entities, violet: shared entities with different emphasis; pink: DePICT-only entities**

### *2.2.4.2 PROV-DM*

*"The term 'provenance' refers to the sources of information, such as people, entities, and processes, involved in producing, influencing, or delivering a piece of data or a thing in the world. In particular, the provenance of information is crucial in deciding whether information is to be trusted, how it should be integrated with other diverse information sources, and how to give credit to its originators when reusing it. In an open and inclusive environment such as the Web, users find information that is often contradictory or questionable: provenance can help those users to make trust judgments."*

(W3C, 2011a)

Provenance metadata is an important category of digital preservation metadata, since its reliable application enables consumers of digital information objects to judge their degree of authenticity if they have undergone change over time. PROV-DM (the provenance data model) (W3C, 2011a) and PROV-O (the provenance ontology) (W3C, 2011b) are an effort by a W3C official Working Group to create a core data model for provenance as a provenance interchange model across

systems. It is generic and domain agnostic. Individual systems can implement their own native domain and application specific representations of provenance and can translate them into the interchange model for information exchange. The model includes the entities depicted in Figure 13.

Entity is a representation of a characterised thing. ProcessExecution represents an activity that has an effect on entities by generating or using them. Agent represents a particular entity that can be associated to activities and is capable of controlling ProcessExecutions. Qualifiers can be associated with relations. And Annotations are used to provide additional, "free-form" information regarding any identifiable construct of the model.

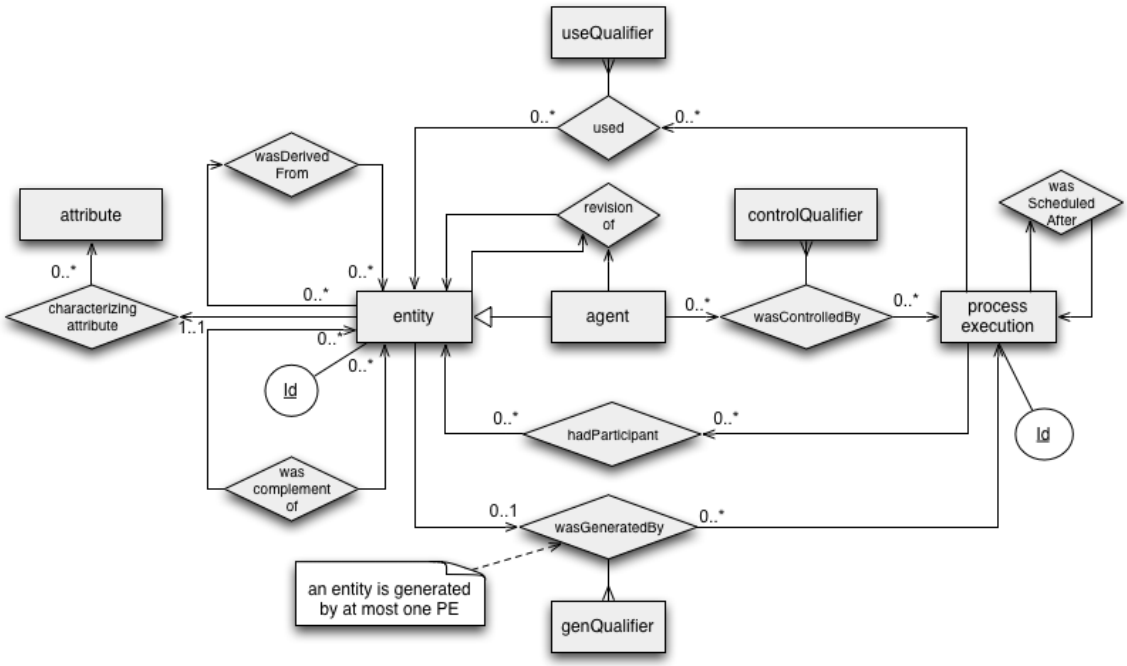



**Figure 13: Entities in Prov-DM (W3C, 2011a)**

**Relationship to DePICT**

As in DePICT, ProcessExecutions (corresponding to *PreservationActions*) are applied to a resource, called Entity (roughly corresponding to *PreservationObject* and/or *Environment*) over time by Agents resulting in derivative Entities or *PreservationObjects* and/or *Environments*. Unlike PREMIS, this covers the whole life-cycle of the digital information object. It models Agents, such as creating applications and message digest originators, explicitly as Agents rather than as Properties that are subordinate to Objects, and, it similarly, models all Events, such as information on the creation event or the granting of rights, as Activities rather than as Properties that are subordinate to Objects. This way of modelling Agents and Events is in line with DePICT's approach. Prov-O's detailed provenance vocabulary can very usefully

instantiate the DePICT model. Unlike Prov-DM, DePICT does not restrict Agents to entities that carry responsibilities, but allows them to take on any kind of role.

#### *2.2.4.3 StratML*

Strategy Markup Language (StratML, nd) is a basic conceptual model for describing the essential contents of a strategy document. It is envisioned as an ISO standardised XML schema and vocabulary for US Federal agency strategic plans that is aligned with the Federal Enterprise Architecture, government policy, and leverages existing standards (StratML, 2006). The non-`Constraint` elements of `Policies` in DePICT can be reused from StratML.

```
<?xml version="1.0" encoding="UTF-8" ?>
<StrategicPlanCore StartDate="1/1/2006" EndDate="12/31/2010" Date="2007-11-27">
  <Submitter FirstName="Owen" LastName="Ambur" PhoneNumber="" EmailAddress="Owen.Ambur@verizon.net"/>
  <Source>http://www.scouting.org/media/strategy/45-016.pdf</Source>
  <Organization>
    <Name>Boy Scouts of America</Name>
    <Acronym>BSA</Acronym>
  </Organization>
  <Vision>The Boy Scouts of America will prepare every eligible youth in America to become a responsible, participating citizen and leader who is guided by the Scout Oath and Law.</Vision>
  <Mission>The mission of the Boy Scouts of America is to prepare young people to make ethical and moral choices over their lifetimes by instilling in them the values of the Scout Oath and Law.</Mission>
  <Goal>
    <SequenceIndicator>1</SequenceIndicator>
    <Name>Opportunity for Involvement</Name>
    <Description>Every Eligible Youth Has an Opportunity to Be Involved in a Quality Scouting
      Experience</Description>
    <Stakeholder />
    <Objective>
      <SequenceIndicator>1.1</SequenceIndicator>
      <Name>Market Share</Name>
      <Description>Increase market share and/or growth.</Description>
      <Stakeholder />
    </Objective>
    <Objective>
      <SequenceIndicator>1.2</SequenceIndicator>
      <Name>New Members</Name>
      <Description>Increase the number of new members.</Description>
      <Stakeholder />
    </Objective>
  </Goal>
</StrategicPlanCore>
```

**Figure 14: An example snippet from http://xml.gov/stratml/BSAStratPlan.xml**

Some top-level elements in StratML as of 2010 are as follows:

- *Submitter*: The person submitting the policy.
- *Source*: The Web address (URL) for the authoritative source of this document
- *Organization*: The legal or logical entity to which the policy applies.
- *Vision*: Vision statements are distinguished from goals in that they are the focus of constant pursuit but can never be satisfied in the sense of being met or completed.
  A concise and inspirational description of a state the organisation will strive to approach over a relatively long span of years but which can ultimately never be fully achieved.
- *Mission*: Mission statement. A brief description of the basic purpose of the organisation. An agency's goals should flow from the mission statement.
- *Value*: A principle that is important and helps to define the essential character of the organisation.
- *Goal*: General goal.
  A relatively broad statement of intended results to be achieved over more than one resource allocation and performance measurement cycle.
  *Goals* define a purpose and direction and take all stakeholders and perceived present and future needs into account. *Goals* must be capable of being effectively pursued with measurable results over more than one budgetary execution cycle but within the reasonably foreseeable future. *Goals* should be objective, quantifiable, measurable, and defined at the level to be achieved by a program activity.
  Supports *Mission*
- *Objective*: Performance goal.
  A target level of results expressed in units against which achievement is to be measured within a single resource allocation and performance execution cycle.
  Supports *Goal*.
  *Objectives* are measurable subsets of *Goals* to be achieved within a given time period with available resources. *Objectives* provide the day-to-day support for achieving *Goals*.

*Submitter*, *Source*, *Organization*, *Vision*, *Mission* and *Value* are adopted in DePICT from StratML.

Within DePICT, these concepts are used in the following way:

- *StratML:Value*, which expresses an (ethical) value of a stakeholder, is different from the "*DePICT:Value*", which expresses the *Value* of a *Characteristic* ( assigned or derived *Value*).
- A *StratML:Objective* is roughly equivalent to a *Constraint* in DePICT. In StratML, an *Objective* is represented as a string. In order to support automated preservation planning, however, a machine-interpretable definition of the *Objective* / *Constraint* is needed. This is developed below.

The other StratML elements provide values that can be simply looked up and used by preservation services. Figure 14 shows an example snippet of a StratML document for the Boy Scouts of America.

### *2.2.5 Specific preservation metadata*

Generally applicable, high-level digital preservation metadata specifications, such as OAIS (CCSDS, 2002) and PREMIS (2012) are domain-agnostic and specify core metadata elements. There are also metadata specification efforts that complement and expand these high-level models. They focus on particular aspects, such as describing preservation metadata for provenance, or on particular technologies, such as describing the preservation metadata needed for image files versus text files. They are too numerous to cover comprehensively in this context. But in the following sections, some particularly relevant ones are presented.

**Relationship to DePICT**

They are not directly relevant for the creation of a high-level conceptual model, such as DePICT, but rather they should be used to extend the high-level concepts in it. It is nonetheless worth-while studying them in some detail in order to ensure that they will fit into the model.

#### *2.2.5.1 Content type specific technical preservation metadata*

A branch of specific preservation metadata is metadata that is needed for specific types of digital objects. There are technical metadata standards that define metadata for specific content types, such as raster or vector image, sound, video, text, spreadsheet, or email. Some content type-specific metadata is essential for rendering a digital object representation. For example, it is essential to know the sample rate of digital audio data, or the width, height, and colour depth of an image.

Some file formats enable the capture of technical, and other, metadata within their files, which has the advantage of keeping the files self-descriptive. However, by extracting and storing metadata explicitly one may also benefit. Separate metadata can:

- be kept small and processed efficiently;
- be distributed separately;
- have different access rights and licensing arrangements than the content;
- help to account for the whole life-cycle of digital objects;
- have its description standardised across file formats; and
- be managed and preserved by preservation systems.

Two examples of content type-specific metadata are the ANSI/NISO Z39.87 (ANSI/NISO, 2006) standard and the textMD (TextMD, nd) specification.

The ANSI NISO Z39.87 standard, Data Dictionary - Technical Metadata for Digital Still Images (ANSI/NISO, 2006), defines semantic units to describe digital raster images. The standard does not prescribe a serialisation. But, in partnership with NISO, the Library of Congress maintains an XML-Schema called MIX (Metadata for Images in XML Schema) (MIX, nd) that is widely used by content creators and in the digital preservation community. Tools, such as JHOVE (JSTOR and the Harvard University Library, nd; Donnelly, 2010; Abrams, Morrissey, Cramer, 2009), are available to extract technical metadata from image files and export the metadata as MIX serialisation.

Like the Z39.87 standard, MIX defines four sections of metadata:

- Basic Digital Object Information: Basic non-content type-specific metadata such as file size, checksums, and format information.
- Basic Image Information: Metadata that is required to render an image, including the compression algorithm and the image dimensions.
- Image Capture Metadata: Metadata about the image capturing process, such as the scanning device, settings and software used in the process.
- Image Assessment Metadata: Metadata important for maintaining the image quality. Information in this section is necessary to assess the accuracy of output. This includes colour information (such as white points and colour maps) and resolution information.

textMD (nd) is a technical metadata specification for text-based digital objects expressed as an XML schema. The schema provides elements for storing the encoding and character information such as byte order, line terminators, character set and information about the technical environment in which the text was created.

It may also store information about the technical requirements for printing or rendering the text on screen. This includes information about sequences and page ordering and may therefore overlap with information stored as structural metadata in the metadata container. While textMD is attached to text files, individual document pages may additionally be defined as distinct objects with their own metadata.

**Relationship to DePICT**

These specific metadata categories fit seamlessly into the DePICT model to extend its vocabulary.

#### *2.2.5.2 Preserving computing environments*

Preservation metadata for computing environments is the information that is needed in order to successfully redeploy computing environments in the future. Metadata for digital objects' computing environments constitutes essential representation information that is needed in order to be able to use digital objects and to make them understandable in the future. This is why metadata about computing environments must be preserved together with the digital objects as part of their core metadata. Furthermore, software components themselves may be the primary objects of preservation, and require a metadata description.

Environments correspond to the "Representation Information" of the OAIS information model (CCSDS, 2012). Representation information is "the information that maps a Data Object into more meaningful concepts" (CCSDS, 2002). Examples for a specific .docx file would be its file format specification that defines how to interpret the bit sequences, a list of software tools that can render it, hardware requirements, the language in which the contained text is written, and context information that states the author, purpose and time of its writing. Environments include documentation, manuals, underlying policy documents, cheat sheets, user behaviour studies, and other soft aids for interpretation.

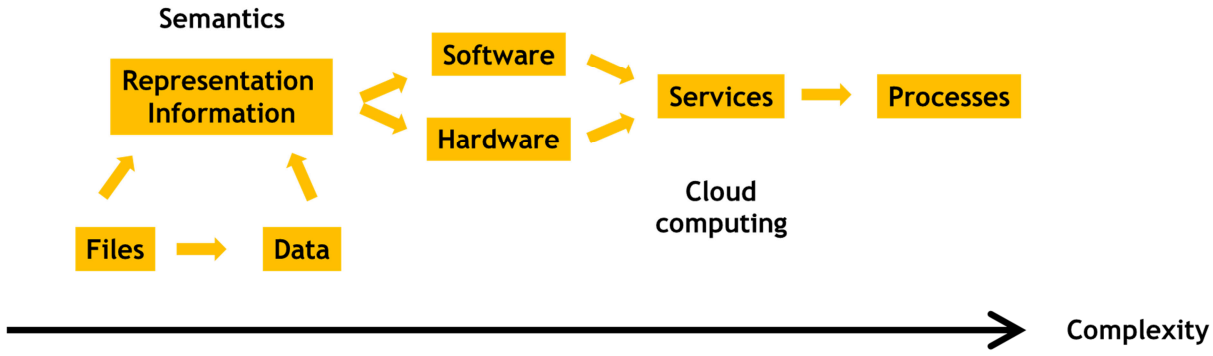



**Figure 15: Increasing complexity of *Environments***

The digital preservation community has initially dealt with less complex objects, such as image files or documents, followed by scientific data held in databases and basic representation information that captures the semantics of files and data so that they can be interpreted in the long-term. Obviously, these "simple" digital objects sometimes are actually quite complex, especially if they contain embedded and linked digital objects or if there is a multitude of rare or customized file formats. Similarly, the representation information quickly becomes very complex since it includes dependencies, such as the underlying software and hardware that are needed to access a digital object. The situation becomes even more complex if there are dependencies on third parties, such as in service and licensing models, for example in the Cloud, where data and software may be outside the repository's immediate control, or if there are distributed computing environments. Processes may be part of a preservation object's representation information. For example, the software and processes that produce scientific data and the scientific data itself are both provenance and representation information for the derived scientific publication, and need to be preserved to provide evidence of its authenticity. Figure 15 illustrates this.

Adrian Brown (Brown, 2008) compiled an overview over Representation Information Registries as of 2008. But much work has been done in the area since. There are several efforts in the digital preservation community to specify the metadata needs for certain aspects of computing environments which are discussed in the following sections.

#### *2.2.5.2.1 Environments as core preservation metadata*

In version 2 of the PREMIS Data Dictionary (PREMIS, 2012), there are four key entities that need to be described to ensure successful long-term preservation of digital objects: Object, Event, Agent and RightsStatement. The Object entity provides two relationships to subordinate environments. For one, there is the Environment semantic unit that permits the description of software, hardware and other dependencies. Rather than being an entity of its own, an Environment is modelled as a semantic unit container that belongs to an Object and is, therefore,

subordinate to the Object entity. The second environment-related semantic unit is the creatingApplication that also is subordinate to the Object entity. Creating applications are outside the scope of an OAIS repository (CCSDS, 2002) and have therefore been historically treated separately from other Environment descriptions. In a generic digital preservation model that is not restricted to OAIS use, but supports the end-to-end digital preservation life-cycle, one would describe Environments uniformly, no matter in what context they are used.

Its subordinate position to Objects means that Environments are only captured to describe an Object's computational context. This has the following limitations:

- Environments are too complex to be handled in an Object repository.
- Environments are rarely specific to a single Object, resulting in their redundant spread across different Objects. These results in
    - unnecessary verbosity;
    - cumbersome management of Environment descriptions as they evolve.
- They are unable to refer to external dedicated registries, which would enable the delegation of "up-to-date and complete" information to an external source if needed.
- They are unable to describe stand-alone Environments and unable to be used for modelling an Environment registry that describes Environment components without the need for creating Objects.

The concrete PREMIS realisation of Environments had further short-comings that should be avoided in a comprehensive model:

- They are primarily applicable to computing environments and do not include representation information in the broader sense, such as documentation, manuals, or applicable software licenses. This restricts the description to a technical level rather than to a level that comprehensively enables redeployment.
- No explicit possibility to document the nature of dependencies between Environments, for example between an operating system and the associated hardware: Is it the only, a possible or a native operating system?
- No links to registry descriptions other than file formats. The model should enable registry references for any type of Environment.

- Specification of versions only for software. The model should enable version information for any type of Environment.

A use case analysis identified the six desirable relationships illustrated in Figure 16. Because Environments are subordinate to Objects, it is impossible to express the latter four of them in PREMIS 2.

1. An Object specifies its Environment, i.e. its computational context. This is the existing relationship in PREMIS 2.
2. An Environment (for example, software source code) is to be preserved in its own right. It is described as Environment and takes on the role of an Object.
3. An Environment takes the role of an Agent (for example, as software Agent involved in a preservation action Event).
4. An Environment is related to another Environment through inclusion, dependency, derivation or other relationships.
5. An Environment has an Event associated with it (for example, a creation or versioning Event).
6. An Environment has an RightsStatement associated with it (for example, license restrictions for a software Environment).

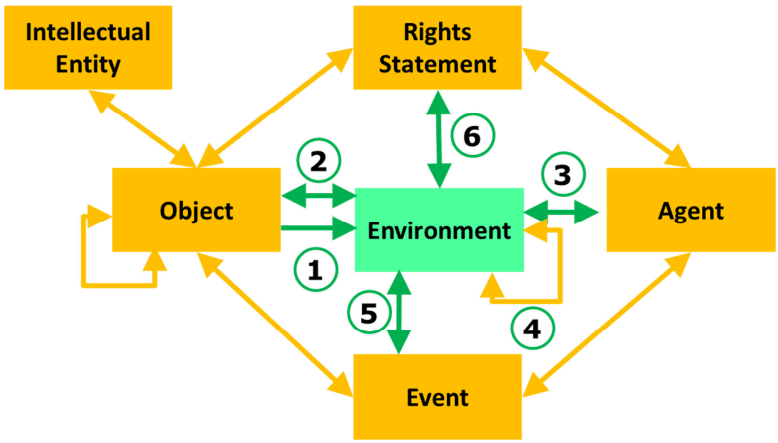



**Figure 16: The basic entities of the PREMIS Data Dictionary (in blue) with the desired Environment entity and their relationships proposed in DePICT.**

The identified shortcomings may be one reason for the fact that the Environment semantic container in PREMIS is rarely used.

**Relationship to DePICT**

Treating Environments as top-level entities, as in the DePICT approach, means that the model can accommodate the requirements described in this section and it can be used for Environment registries that describe Environment components without the need for creating *PreservationObjects*. Treating them as top-level entities also makes it more natural to model *PreservationActions* that directly impact *Environments*, such as data carrier refresh or emulation, as easily as *PreservationActions* that directly impact *PreservationObjects*, say migration. While describing those actions is possible with the PREMIS model, it comes less natural.

The goal of the PREMIS Environment Working group is to rethink the metadata specification for environments. Their description must meet the improved understanding of how to ensure their longevity.

#### *2.2.5.2.2 Software preservation metadata*

Software is a specific content-type that needs to be preserved. In recent research on metadata for software or computer games preservation it is very obvious, that in those domains, non-technical metadata is very important. This includes, for example, functional descriptions, user descriptions, and the legal and licensing context. They, together with the software and hardware dependencies need to be modelled as *Environments* and their *Properties*. The Preserving Virtual Worlds Project (McDonough et al., 2010), an on-going initiative on preserving games and interactive fiction, for example, identified a multitude of necessary preservation metadata: because of the social complexity of a game, client and server preservation alone does not tell the game's story line; rather, how the game is used and appropriated by the players also needs to be preserved. Copyright law requires the capture of every rights holder's consent for preservation. In sites created through social networks, such as "Second Life" (Second Life, nd), this includes every contributor to the site. All the standards for the computer languages, file formats and data types involved need to be preserved to ensure recreatability of the game.

The team used the Functional Requirements for Bibliographic Records (FRBR) model (FRBR, 1998) to capture work, version, variant and implementation information for individually licensed copies. The resulting ontology captures these concepts in hierarchical trees and the relationships between these components. They used the OWL representation language (McGuinness, van Harmelen, 2004; W3C OWL Working Group, 2009) to relate the 2 data models of FRBR and OAIS (CCSDS, 2002), created an ontology, and used the Metadata Encoding and Transmission Standard

(METS) (METS, nd) to package up the information. The goal of the ontology is to be able to collect enough information so you could write an emulator for the game if you have none.

"The Significant Properties of Software: A Study" (Matthews et al., 2008; Software Sustainability Institute, Curtis+Cartwright, nd) also uses FRBR as the guiding framework, defining Functionality, Provenance and Ownership, Software Environment, Software Architecture, Operating Performance, Software Composition and User Interaction metadata on each of the four levels of Package, Version, Variant and Download. Further examples of metadata models and ontology proposed for software preservation can be found in SWOP: The Software Ontology Project (SWO, nd; SWOP, nd) and DOAP (Dumbill, nd). Karsten Huth proposed a metadata model for preserving computer games in his Master's thesis (Huth, 2004). The POCOS project (POCOS, nd) investigated metadata issues related to the preservation of complex objects, such as Visualisations and Simulations, Software Art, and Gaming Environments and Virtual Worlds.

**Relationship to DePICT**

These metadata investigations provide valuable substructures for DePICT's *Preservation-Object*, *Agent* and *Environment* entities.

#### *2.2.5.2.3 Technical environment metadata registries*

OAIS repositories are different from registries. While OAIS repositories need to describe concrete *PreservationObjects* held in their care, registries often capture generic information that can be reused in specific instances. Often the former links to information held in the latter. Since registries are generic they do not hold context or organisation specific metadata. The TOTEM scalable, generic metadata schema for documenting technical *Environments* (TOTEM, nd; Delve, 2011; Delve, Anderson, 2012) was developed within the KEEP project on emulation (KEEP, 2012). It models complete, technical computing *Environments* in order to facilitate uptake of emulation as a digital preservation approach. Readily available descriptions of technical *Environments* help organisations "identify, describe and manage the documentation of technical environments in a systematic and consistent way that meets emulation, data management and archival requirements." (Delve, 2011). Since software forms part of a computing environment, TOTEM includes technical software preservation metadata, as discussed in the previous section, but does not include their non-technical aspects in its scope. The schema currently covers the PC x86 architecture, the Commodore 64 architecture and console gaming platforms. For PC and Commodore *Environments* it describes Files, Software, Libraries, Operating Systems detailed to version level, and Hardware. For Console Games it covers metadata

that describes the Console Game Version, Controllers, and Consoles. It has been implemented as a MySQL database at the University of Portsmouth and offers universal access through a web-service. Data has been obtained from live catalogue data for digital collection objects, from sources such as Mediapedia (Mediapedia, nd) and through document analysis. This work is part of a recent move towards building registries for describing the technical properties of computing and gaming environments including software and hardware components. Similarly, the IIPC (IIPC Preservation Working Group, nd) has developed a technical database using a computing environment schema as a foundation for web archiving, and TOSEC (short for "The Old School Emulation Centre") (TOSEC, nd) "is dedicated to the cataloguing and preservation of software, firmware and resources for microcomputers, minicomputers and video game consoles."

Examples of software registries, the NSRL National Software Reference Library (NSRL, nd), MobyGames (MobyGames, nd) and AMINET (Aminet, nd), illustrate practically used metadata schemas, but do not necessarily support digital preservation functions. Jhove (JSTOR, Harvard University Library, 2012; Abrams, Morrissey, Cramer, 2009), PRONOM (Brown, 2005; TNA, nd), UDFR (nd-a) and the Library of Congress (Library of Congress, nd-a) have defined metadata that is needed to technically or qualitatively describe file formats and have built registries based on these metadata definitions. This includes some software metadata specifications, which, for PRONOM, are now available in a linked data representation and for UDFR contains software description in the recently released UDFR database (UDFR, nd-b).

**Relationship to DePICT**

DePICT's model, unlike, for example, PREMIS, covers both registries' and repositories' functionality. This means that it comprehensively covers technical, organisational and context-dependent modelling elements. As OAIS repositories have a tendency to model characteristics of the `PreservationObjects` themselves in favour of modelling the complete representation information including their computing `Environments`, they inherently offer less support to emulation solutions. Technical registries counter-balance this tendency. Just like for software preservation metadata approaches discussed above, these metadata investigations provide valuable substructures for DePICT's `PreservationObject`, `Agent` and `Environment` entities.

#### *2.2.5.2.4 Metadata for virtualised infrastructures*

Metadata that is used to capture complex dependencies is addressed in initiatives such as VRDF (Kadobayashi, 2010). The VRDF project develops a framework to describe and analyse complex dependencies of services, virtual machines, virtual routers and VLANs of virtualized

infrastructures in order to improve the availability of data centres and maintain their security level. Their RDF schema enumerates the virtual and physical resources and describes the dependencies between them. Other examples are the Cloud Data Management Interface (CDMI) (SNIA, 2012) which "describes the functional interface that applications use to create, retrieve, update and delete data elements from the Cloud"; and the Web Service Definition Language (WSDL) (Christensen, Curbera, Meredith, Weerawarana, 2001), which describes network services as a set of endpoints operating on messages.

**Relationship to DePICT**

Again, these metadata investigations provide valuable substructures for DePICT's `Environment` entity.

#### *2.2.5.2.5 Business process preservation*

The TIMBUS project (Edelstein et al., 2011; TIMBUS, nd), a 3-year EU co-funded project has an even wider scope than discussed in the previous sections. It addresses the challenge of digital preservation of business processes and services to ensure their long-term continued access. "This approach extends traditional digital preservation approaches by introducing the need to analyse and sustain accessibility to business processes and the supporting services, and it aligns preservation actions more fully with enterprise risk management (ERM) and business continuity management (BCM)." (TIMBUS, nd).

TIMBUS analyses and recommends which aspects of a business process should be preserved and how to preserve them. This task centres on the identification of the necessary preservation metadata to be able to redeploy processes, services, and computing environments and their dependencies or the functionality afforded through them, and the data used in them. It delivers methodologies and tools to capture and formalise business processes on both technical and organisational levels. This includes preservation of their underlying software infrastructure, virtualisation of their hardware infrastructure and capture of dependencies on local and third-party services and information. This means that, in addition to technical preservation metadata, it draws on metadata standards that capture business processes and is identifying forms of supporting business documentation needed to redeploy processes and services.

Examples of modelling languages for the business processes themselves are Archimate (Open Group, 2012), BPMN (Object Management Group, 2011a), Petri Nets (Hillah et al., 2009) or, for software workflows, workflow engines, such as Taverna (Taverna, nd), Kepler (Ludäscher et al., 2006) and Activiti (Activiti, nd). Research efforts are under way to capture dependencies between

*Environment* components. These can be dependencies between services, software and libraries, software and hardware, etc. For example the Debian operating system (Debian, nd) defines a vocabulary of possible package dependencies, where a package is the minimum amount of software to provide a certain function that can be reused for digital preservation purposes. TIMBUS is, at the time of writing, just starting to develop their ontology to describe this domain.

The commercial imperative for business process preservation comes from several pressures. Heavily regulated industries, such as pharmaceuticals and aircraft manufacture must fully document processes so that they can be audited, reproduced, or diagnosed. This provenance information may be the basis in the case of litigation. Long-lived companies must manage services across multiple changes in technical environments; they may need precise process specifications to reproduce the same functionality on a new platform. If processes are outside an organisation's control, such as in service and licensing models, especially in the Cloud, they may use an escrow service in order to mitigate the risk of losing access to the data or services they depend on. They must be confident that all of the needed information is demonstrably included in the escrow agreement and services. This problem is isomorphic with the digital preservation of software. Organisations undergoing major staff changes must ensure that they retain the knowledge needed to operate or re-instate production processes.

In addition to publications and data, academics need information about the software and processes that produced them to assess the validity of the data and the derived scientific claims. The same provenance information that can provide a key in regulated industries can also support credit assignment in academia. Process information provides a form of provenance metadata, which documents stewardship and the events that have impacted the resulting process products. This information is generally important to prove the authenticity or quality of process products. All industries benefit from analysis of processes that may lead to their continuous improvement.

**Relationship to DePICT**

The challenge for TIMBUS is to define the right metadata schemas to capture all process components necessary for process redeployment. These process descriptions, although being very complex digital objects, can be treated in the same way as other *PreservationObjects* or *Environments* in DePICT.

## 2.3 Constraints on preservation objects, environments and preservation actions

One of the key concepts in digital preservation is that of *Constraint* that defines limitations or restrictions on the space of allowable preservation actions. *Constraints* make the stakeholder's values explicit and influence the digital preservation process. *Constraints* are often expressed in *Policies*, and refer to *Characteristics* of *PreservationObjects*, *PreservationActions* or *Environments*.

### 2.3.1 Significance constraints

The most common form of *Constraints* discussed in the digital preservation literature is *SignificanceConstraints*, also called significant properties (for example, Hockx-Yu and Knight, 2008; Knight, 2008; Knight, Pennock, 2009), significant characteristics (Thaller et al., 2008; Becker et al., 2008a), essence (NAA, 2008), aspects (Clausen, 2007), and others. Original work on *SignificanceConstraints* comes out of the Cedars project (CEDARS, nd), work at the Australian National Archives (NAA, 2008), the InSPECT project (Knight, 2008), Planets (Becker et al., 2008a; Becker et al., 2008b; Clausen, 2007; Dappert, 2009; TNA, nd) and others. Comprehensive surveys of related work in this area are provided by Knight (Knight, Pennock, 2009) and Wilson (2007).

They specify, as business *Constraints,* "the characteristics of digital objects that must be preserved over time in order to ensure the continued accessibility, usability, and meaning of the objects, and their capacity to be accepted as evidence of what they purport to record." (Wilson, 2007). Section 3.2.3.2.2 discusses them in depth.

The term "characteristics", which describes what must be preserved in this definition, is interpreted in two conflicting ways. Some interpret it to refer to the abstract properties of file formats (e.g., Becker et al., 2008a; Knight, 2008), whereas others interpret it to refer to the values of properties of specific digital objects (Becker et al., 2008b). One also finds different interpretations of the term "digital objects", which describes which characteristics need to be preserved. In 2002, the OCLC/RLG Working Group on Preservation Metadata (OCLC/RLG Working Group on Preservation Metadata, 2002) stated that the properties of data objects need to be preserved; Brown (Brown, 2008) applies it to information objects as opposed to data objects in the OAIS sense of the terms (CCSDS, 2002); Becker (Becker et al., 2008a) applies it to the characteristics of specific file formats. Knight hints that the characteristics of the environments in which digital objects are rendered may also have to be preserved (Knight, 2008), but this idea is

not fully articulated there and is not developed until expressed by Dappert and Farquhar (2009b) and Anderson et al. (2010). Chris Rusbridge (Rusbridge, 2006) eloquently states why the quest for faithfulness to the original in all respects is both excessive and impractical in most preservation situations. The need to clarify the difference between *SignificanceConstraints* and representation information has repeatedly been voiced (e.g., Hockx-Yu and Knight, 2008; Knight, Pennock, 2009), but not previously addressed.

**Relationship to DePICT**

All previous treatments on *SignificanceConstraints* limit themselves to the identification of *Properties* they consider significant either for certain file types or content types. This means that there are no allowances for tolerances, for specifying the relative importance of different *Constraints*, for specifying pre- or post-conditions, for specifying *Characteristics* in the relationship between *PreservationObjects* rather than simply on one *PreservationObject* or for specifying *SignificanceConstraints* on *Environments*. DePICT defines *Constraints* based on the requirements identified in the analysis of the digital preservation domain, addresses the gaps identified in the analysis, clearly defines the terminology and relates them to the DePICT conceptual model to avoid vagueness.

### *2.3.2 Preservation planning constraints*

In a practical application, the Plato tool (Becker et al., 2008b) uses *SignificanceConstraints* of *PreservationObjects* together with other *Constraints* to guide the preservation planning process. The Plato planning tool is a "decision support tool that implements a solid preservation planning process and integrates services for content characterisation, preservation action and automatic object comparison in a service-oriented architecture to provide maximum support for preservation planning endeavours. PLATO is a web based tool to help librarians, archivists, and curators weigh alternatives and decide which, if any, preservation actions to undertake for a specific set of records." (Prom, 2010).

Plato's decisions are based on the Requirements underlying the planning process. These Requirements have the role of *Constraints* and are expressed as propositional statements in free-text. In order to help the requirements gathering process, they are organised in hierarchical Objective Trees with a high-level structure suggesting the break-down into Object Characteristics ("Content", "Context", "Structure", "Appearance" and "Behaviour"), Record Characteristics describing the digital record, Process Characteristics describing the preservation process and Cost. But the tree structure has limited impact on the reasoning. The leaves of the tree branches are,

ideally, measurable and comparable criteria but may resort to subjective scales. It is possible, at this point to define the desirable *Value* of an extractable *Characteristic* rather than just a propositional statement.

**Relationship to DePICT**

Plato *Constraints* are fundamentally propositional and do not tie into a conceptual model. The emphasis, in Plato, is on the decision making process, rather than on the integrated conceptual modelling of the domain. Recently, Plato suggests that *Characteristics* are structured as *Object* and *Process Characteristics*, but this distinction is only to organise the requirements gathering, rather than to structure the domain conceptually.

A propositional reasoning system is less expressive than the parameterised *Object-Property-Value* model used in DEPICT. This means that concepts, elements and attributes of the Plato systems can be mapped onto the DEPICT model, but not necessarily the other way round.

## 2.4 Functional components of digital preservation

A conceptual model must be tested on the use cases of its domain. DePICT was, amongst others, validated in the field against work-packages in the Planets project (Farquhar, Hockx-Yu, 2007; Planets, nd), the SCAPE project (Edelstein et al., 2011; SCAPE, nd) and the TIMBUS project (Edelstein et al., 2011; TIMBUS, nd). It was also validated against the digital preservation functional descriptions in the OAIS (CCSDS, 2009) and Planets (Sierman, 2009) models that are described in this section. All of chapter 4 is dedicated to analysing the suitability of the DePICT model for digital preservation use cases. Many more related systems and research activities will be introduced in detail there.

### 2.4.1 OAIS

As mentioned in section 2.1 and illustrated in Figure 17, the OAIS model (CCSDS, 2002) comprises a functional framework consisting of 7 types of functional preservation services: Ingest, Archival Storage, Data Management, Administration, Preservation Planning, Access and Common Services supplied by any IT system. They contain a high level of refinement.

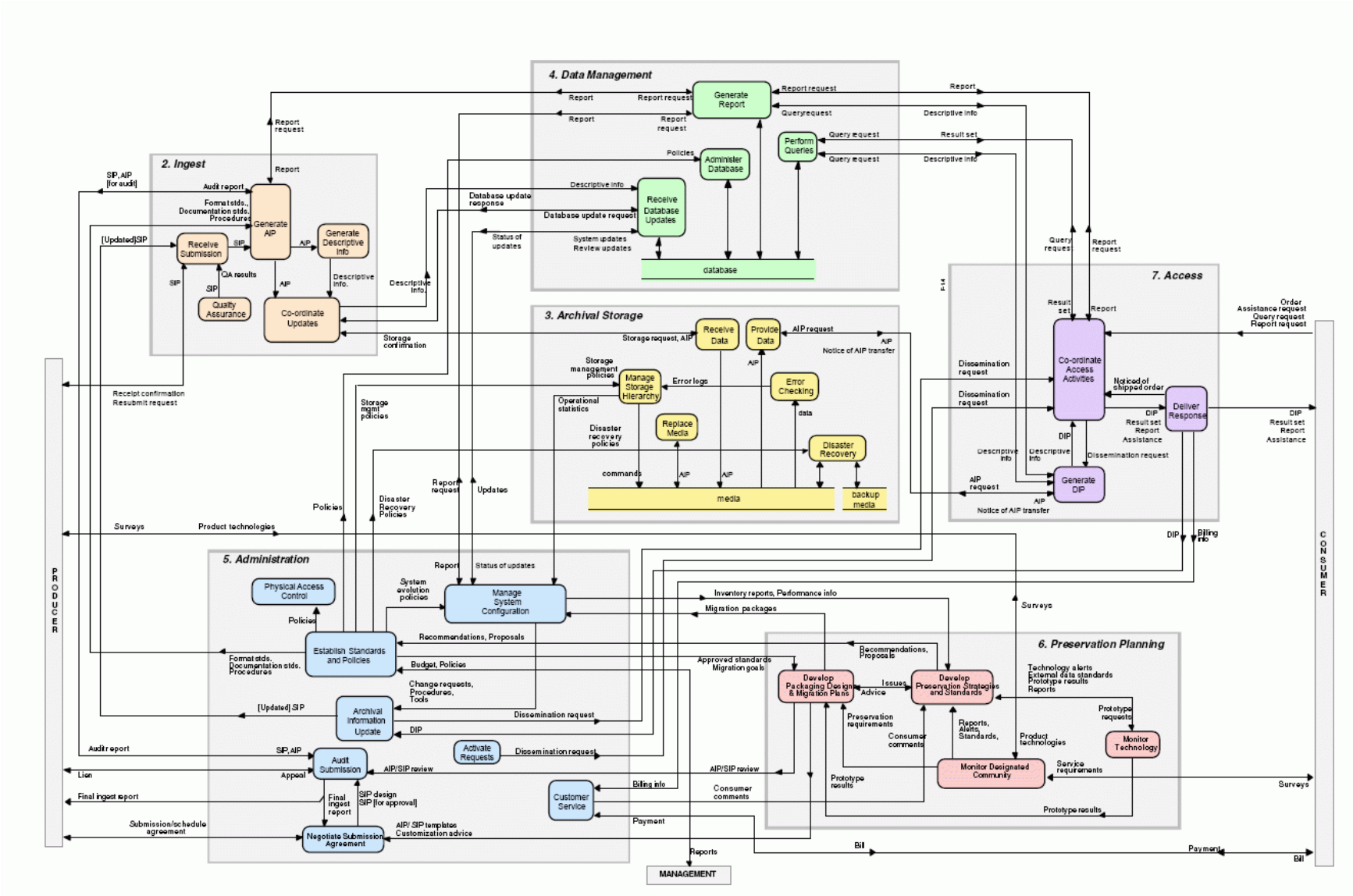



**Figure 17: Composite diagram of OAIS functional entities (MathArc, nd)**

### *2.4.2 The Planets functional model*

The Planets PP7 work-package (Sierman, 2009) describes the relationship between the Planets and the OAIS preservation planning processes. The comparison shows that in the Planets Model Preservation Watch is modelled more comprehensively; Risk Management is an integral part of the process; Emulation is a fundamental preservation methodology; Characterisation plays an important role as a basis for Preservation Planning, for the preservation process itself and for validating the success of preservation actions, and the capturing and recording of Representation Information is well embedded in the Preservation Planning process.

**Relationship to DePICT**

Both the Planets and OAIS functional models have been analysed in DePICT. Their identified functions have been categorised by the role they play in metadata creation, storage, evaluation and exchange. This ensures a seamless flow of information from function to successor function in the whole life-cycle of `PreservationObjects`. This analysis is discussed in detail in section 4.

## 2.5 Risk management

Digital preservation has been defined by Jones and Beagrie as “The series of managed activities necessary to ensure continued access to digital materials for as long as necessary” (Jones, Beagrie, 2001/2008). These managed activities fall into two categories (Dappert, 2011). The first one is to keep risks to digital assets from becoming issues (risk driven, proactive preservation as risk management tasks), and the second one is to deal with issues when they have arisen (remedial, reactive restoration). Risks are defined as uncertainties of outcome - things that may happen. Mostly they are seen as threats with negative impact, but they can also be opportunities with positive impact. In contrast, issues have appeared and might have to be remedied. The above definition implicitly states that digital preservation is inherently a risk management activity, as presupposing that we need to *ensure* continued access implies that there exist risks of losing this access. And this refers to the first of the above mentioned activities.

Risk management is a well-established methodology for identifying and assessing risks and for identifying and prioritizing actions that mitigate them. Standards include, for example, the ISO 31000 (ISO, 2009) family of standards codified by the International Organisation for Standardisation that defines risk management principles, a framework and the general process. More specific standards for information risk management, tailored to the information asset industry, exist, such as Control Objectives for Information and related Technology (COBIT) (IT Governance Institute, 2007) and ISO/IEC 27000 (ISO, 2005). Guidelines are often issued by national governments (e.g. HM Government, 2008). Additionally, there are information risk management guidelines specific to the longevity of digital assets (e.g. TNA, 2011).

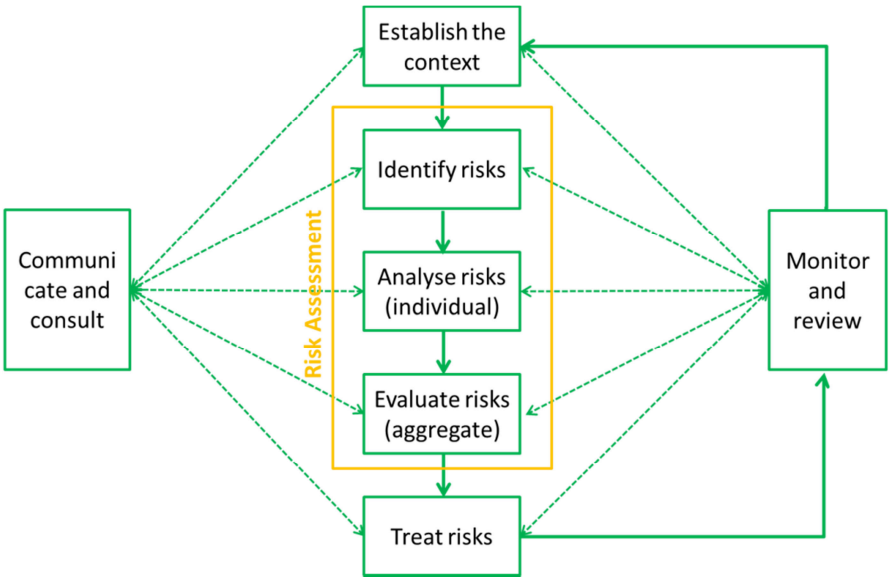



**Figure 18: ISO 31000 risk management process**

The general risk management process, as defined in ISO 31000, is illustrated in Figure 18. Communication and consultation are central to every step, since risk management is an information-centric task. The quality of the decisions taken is dependent on the quality of the information that they are based on. Risk management professionals do not have sufficient information to identify risks and to make informed decisions on their own: they need to collaborate closely with management and stakeholders. The first step in the iterative process is establishing the context in which risk management is performed. This includes identifying scope, objectives, goals, decision makers, stakeholders, assets, the legal and regulatory and technological framework, etc. Once the information is gathered, one can perform Risk Assessment (depicted in the yellow box). It consists of three steps: identifying individual risk; analysing each individual risk as to its impact and severity; then evaluating all risks relative to each other. The next step identifies mitigating actions and the required resources and goes on to prioritize them. The final step is the risk treatment step in which the risk is mitigated. All steps require on-going monitoring and review and have to be repeated in defined intervals.

Digital preservation processes fit well into and can be mapped onto this process. Canteiro and Barateiro (2011), for example, map the digital preservation of e-Science data to this process. The workflow of the Plato preservation planning tool (TU Wien, nd-a) mimics the ISO 31000 process without stating so explicitly. For example, for establishing the context, Plato describes the collection under consideration and collects the constraints that guide digital preservation in so-called requirements trees. For risk assessment it defines thresholds and tolerances for each constraint. In the planning step it looks at all preservation action options and prioritises them based on the previous analysis, and suggests the most suitable action. The DRAMBORA (Digital Repository Audit Method Based on Risk Assessment) interactive online-tool (McHugh, Innocenti, Ross, 2008) implements the ISO 31000 process. By structuring the process around functions relevant to digital preservation and by developing a risk ontology derived from risks that were asserted in previous uses of the tool, it tailors it to the risk assessment of long-term aspects of digital collections. Barateiro and Borbinha (2011) have developed a formal definition of risk management concepts and have implemented it as XML-based domain specific language for risk management (Risk-DL).

**Relationship to DePICT**

In spite of this obvious connection of risks to digital preservation, existing digital preservation conceptualisations do not include the PreservationRisk entity in their models or permit recording risks as provenance information and drivers for preservation actions.

# 3 The conceptual model and its requirements

This chapter describes the key concepts of the digital preservation domain of the DePICT model. In each section, a new entity with its properties and relationships is introduced. This is supported by vocabulary for possible sub-classes of the primary entities and by requirements on the functionality that needs to be supported by digital preservation services. Organisations can create sub-classes that are suitable for the organisations' contexts, inherit the functionality from the DePICT super classes and customise additional functionality for the newly created sub-classes.

This chapter captures important observations about the concepts in the digital preservation domain that need to be considered in a formal description of the domain in order to support all digital preservation functionality and to enable manual or automatic digital preservation reasoning. Appendix 7.1 translates these informal observations into a formal model. It investigates the specific properties of the concepts. It illustrates these properties with examples, but makes no effort to compile a complete ontology or vocabulary for sub-classes of the key concepts or for permissible values. There are many separate efforts in the digital preservation community with this goal (for example NCBI & NLM, 2012; FRBR, 1998).

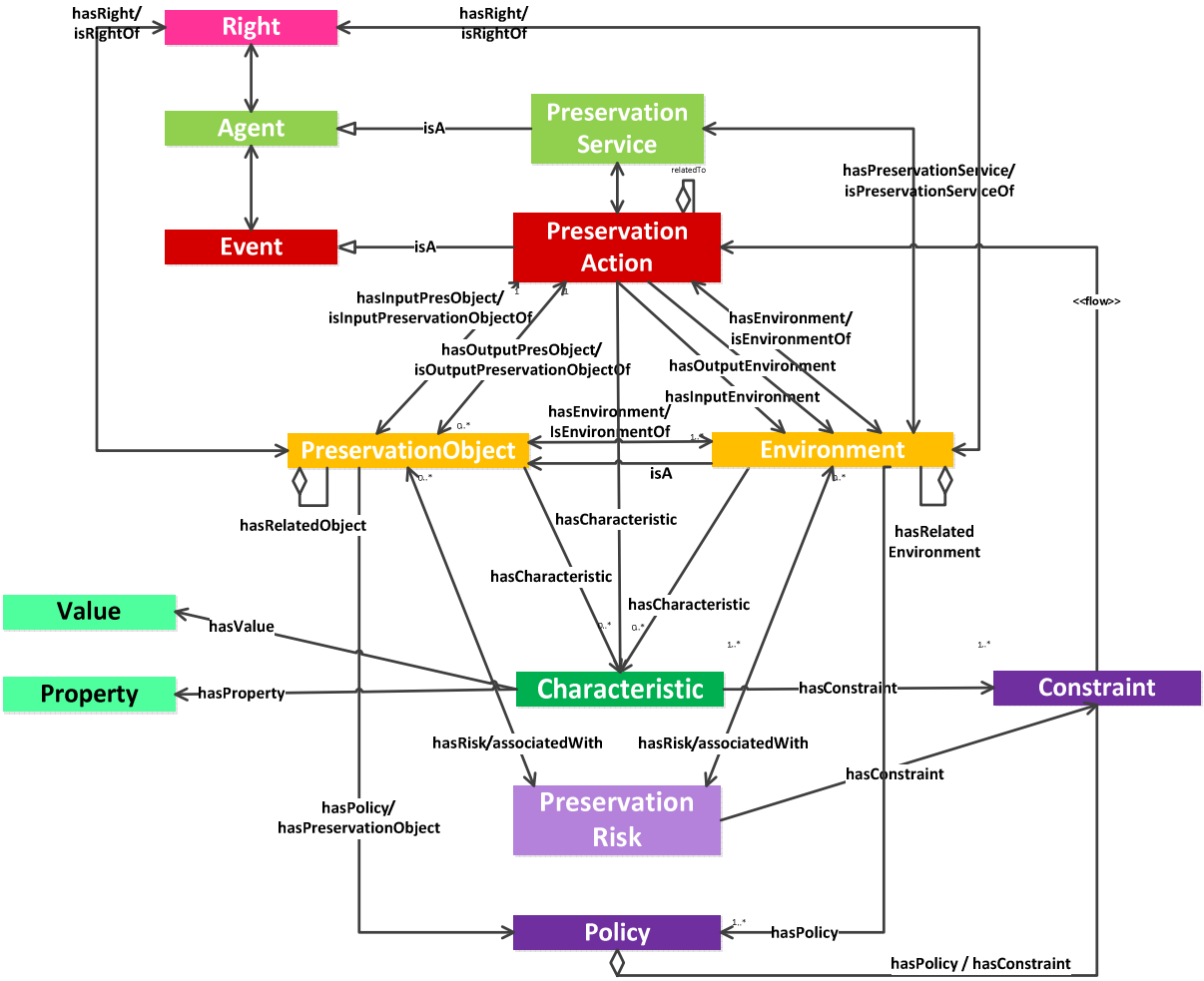



**Figure 19: The conceptual model for digital preservation**

For ease of readability, section 3.1 describes a core conceptual model for digital preservation that supports a static view of the preservation environment. Section 3.2 will add to this to create a complete view of the concepts needed to describe the digital preservation domain in order to support dynamic interactions of *Preservation Services*. Please refer to the complete view shown in Figure 19 while the model is being explained incrementally. It is sometimes necessary to refer to entities in the figure before they have been properly introduced. For readability, the figure captures only the most important relationships. For a comprehensive description please see the conceptual detail defined in appendix 7.1.

## *3.1 The core conceptual model*

The core conceptual model captures the objects of preservation and their characteristics - the core entities necessary to describe the things that need to be preserved in the long-term. The entities are depicted in Figure 20. In summary, any *PreservationObject* has one or more *Environments*. Every *PreservationObject* may be decomposed into related sub-

*PreservationObjects*. Every *Environment* in which the *PreservationObject* is embedded consists of one or more sub-*Environments*, such as hardware and software environments, the legal system, and other internal and external factors. *PreservationObjects* and *Environments* are described by *Characteristics*. *Characteristics* describe the state of *PreservationObjects* and *Environments* as *Property* / *Value* pairs.

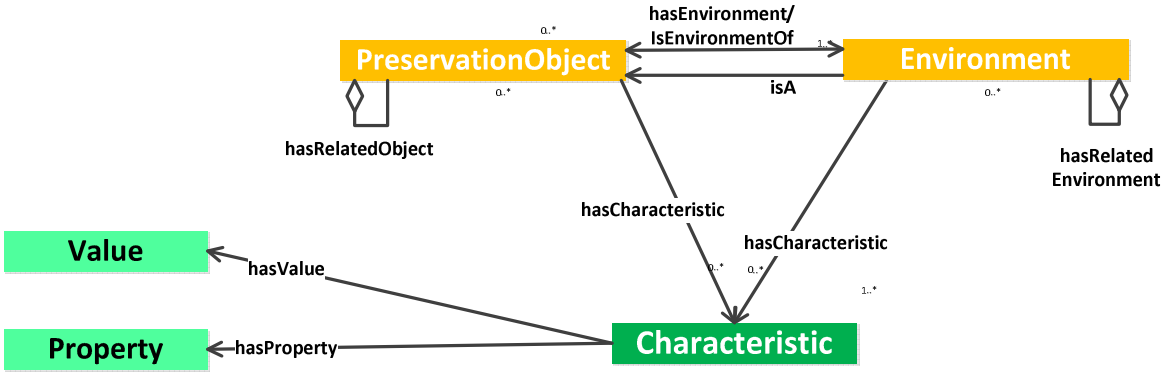



**Figure 20: Core conceptual model**

### 3.1.2 Preservation objects

Definition of PreservationObject

A *PreservationObject* is any object that can directly or indirectly be at risk and needs to be digitally preserved.

Modelling Requirements for PreservationObjects

A *Bitstream* is the primary *PreservationObject*. If it is at risk, it becomes the object of preservation activity. A *Bitstream* is, however, embedded in a larger context, as illustrated in Figure 21. Higher-level objects, such as the *Representation* (one complete rendition of an *IntellectualEntity*) that contains the affected *Bitstream*, and the *IntellectualEntity* which is rendered by this *Representation*, are indirectly affected by its preservation need. They need to be considered during preservation planning and are, therefore, indirectly *PreservationObjects*. Conversely, a stakeholder cannot consider the preservation of each individual data object in isolation. The intellectual content dictates the properties of the *Bitstream* that encodes it and therefore dictates how it should be preserved. Also, stakeholders need to take a global look at all their collections and resources in order to prioritise their *PreservationActions* and co-ordinate preservation activity. Since *PreservationObjects* need to be described at these different levels, the model has *PreservationObject* sub-classes on three tiers, as illustrated in Figure 21.

The top tier comprises physical objects, such as *Bitstreams* and their sub-classes, including *Bytestreams* and *Files*. They are the primary physical *PreservationObjects*. If they are at risk of decay or obsolescence, they become the objects of preservation.

The middle tier comprises *Representations*, which are the set of *RepresentationBitstreams* that are needed to create a single rendition of an *IntellectualEntity*, for example, the set of .html and .gif *Files*[13] needed to render the web version of a journal article.

This model, unlike PREMIS, distinguishes between logical and physical aspects of bitstreams. *Representations* consist of *RepresentationBitstreams*, the logical, ideal bitstream that make up the *Representation*. They may have *Properties*, such as their *targetFileChecksum* and *fileName* and *SignificanceConstraints* that need to be

[13] The formal definition of such a statement would of course contain a persistent unique identifier of the exact version of the file formats. For improved readability of examples the text informally refers to file formats by their file extension.

satisfied during *PreservationActions*. *Bitstreams* are the physical, actual counterparts of *RepresentationBitstreams*. They, for example, may be corrupted and have *observedFileChecksums* varying from those of their *RepresentationBitstreams*. A *RepresentationBitstream* may be implemented by several *Bitstreams* if there is replication present in the system.

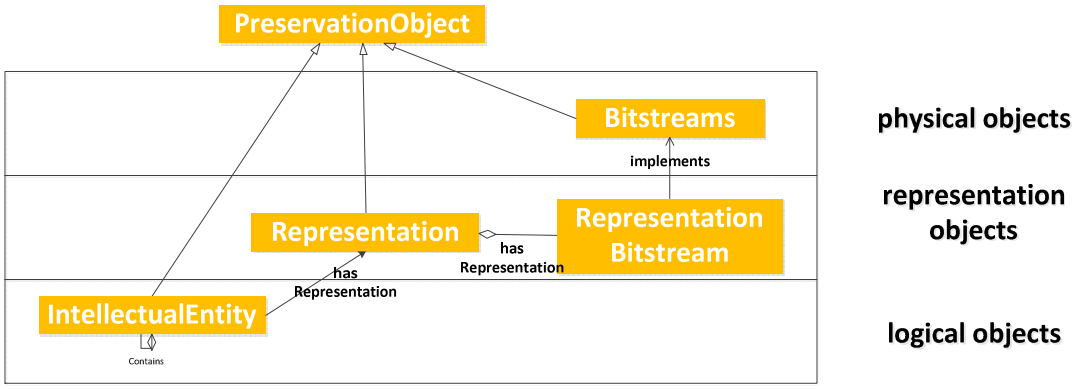



**Figure 21: *PreservationObject* sub-classes in a 3-tiered *PreservationObject* hierarchy**

The bottom tier comprises the logical objects called *IntellectualEntity*. An *IntellectualEntity* is a distinct intellectual or artistic creation that a curator decides is relevant to the Designated Community. PREMIS (2012) defines it as a set of content that is considered a single intellectual unit for purposes of management **and** description[14]. The requirements of the organisation and system to be modelled determine which entities are actually modelled as *IntellectualEntities* and, therefore, the *IntellectualEntity* must be extended in ways to meet the needs of stakeholders.

Most repositories support discovery and delivery of *IntellectualEntities* such as *Books*, *Videos*, and *Articles* for the Designated Community of readers. They may augment these with *Work* and *Expression* sub-classes to capture useful FRBR distinctions (FRBR, 1998). But *IntellectualEntity* may also correspond to larger structures, such as *Collections* (different levels of aggregation), which may not be of interest to the reader, but may be significant to the Designated Community of curators and researchers, and may be relevant in preservation decisions.

During preservation, it is also often necessary to consider fine-grained components of an *IntellectualEntity* that need to be described individually. They may, for example, be modelled in order to capture *SignificanceConstraints* that need to be satisfied during *PreservationActions*. Examples include *Table*, *Image*, *Title*, *Substring*, or even an

[14] *Bitstreams*, *RepresentationBitstreams* and *Representations* are also for the purpose of management but not for the purpose of description.

individual *Character*. *Components* can be classified in several ways, such as by the type of content (e.g., *TextComponent, ImageComponent, TableComponent)*, or by structure (e.g., *HeaderComponent* or *TableOfContentsComponent*).[15] These *Components* themselves are *IntellectualEntities* for the purpose of management and description.

The *Environment* entities, introduced in the next section, can be *PreservationObjects* themselves. For example, software code or a games controller can be objects of preservation, in addition to being part of another *PreservationObject*'s *Environment*. On a level of description and management they can be captured as *IntellectualEntities*. In this case, both physical and digital items are captured through metadata. Software, as a traditional, digital object can be additionally described through *Representations*, *Representation-Bitstreams* and *Bitstreams* as mentioned earlier For physical, non-digital *PreservationObjects* the *Bitstream* level corresponds to individual physical objects that are subject to physical *PreservationActions*. On the representation layer, non-digital *PreservationObjects* can, similar to digital artefacts, have structural descriptions of all the basic physical parts that together render the non-digital *IntellectualEntity* functional, such as a circuit diagram of all components that together make up a games controller. The language in this thesis may not naturally support this interpretation. But the analogy can be made as described here.

Chapter 2 discussed the relationship between DePICT's *PreservationObject* categories and those defined in OAIS (CCSDS, 2012) and PREMIS (2012). The concept corresponding to *Bitstreams* is the Data Object in the OAIS model, but there is no equivalent to *RepresentationBitstreams*, *Representations* and *IntellectualEntities*. The PREMIS data dictionary distinguishes *Files* and *Bitstreams*. PREMIS has restrictions on *Bitstreams* that should not be applied in a general conceptual model. PREMIS does not allow for the distinction between *Bitstreams* (the actual bit sequence) and *RepresentationBitstreams* (the ideal bit sequence). For a detailed discussion see chapter 2.

**Modelling Requirement 1.1.1:**

It is important to realise that the conceptual model can be instantiated in various frameworks, without tying *IntellectualEntities* or any of the other entities to a single one of them. The conceptual model is and needs to be universally applicable and not restricted to a limited

[15] *Values* for *Characteristics* of *Components* can be measured from their associated *Representations* (e.g. the *font* of a *character component* can be extracted from its *RepresentationBitstream*.).

community. For example, in the library setting, *IntellectualEntity* sub-classes may include *Work* and *Expression* to capture useful FRBR distinctions (FRBR, 1998). In an archival setting, sub-classes such as *Fonds* and *Series* are relevant.

**Modelling Requirement 1.1.2:**

In the simplest case, a *Bitstream*, *RepresentationBitstream*, *Representation* and *IntellectualEntity* have a one-to-one correspondence. For example, a book *IntellectualEntity* might be represented as a *Representation* consisting of a single .pdf *RepresentationBitstream* which is implemented in a single *File* in the .pdf format. In other cases, however, several *Bitstreams* may implement one *RepresentationBitstream* and several *RepresentationBitstreams* may make up one *Representation*. For example, the same book *IntellectualEntity* might be represented with one .pdf *File* per chapter, each of which contains an embedded image for each of several pages.

**Modelling Requirement 1.1.3:**

Preservation activities that take place in the context of a content-holding institution, such as a library, involve considerations that go beyond an individual *File* or *Representation*.

Example electronic journal collection:

Consider the case of a library that has a substantial *Collection*, an *IntellectualEntity* for the purpose of management and description:

In this scenario the *Organisation*, its *Collections*, their *JournalTitles*, their *Issues*, and their *Articles* can be types (i.e. sub-classes) of *IntellectualEntities*. The primary logical object of preservation is an *IntellectualEntity* of type *Article*. The article can have several *Representations* that render it with the aid of suitable software and hardware *Environments*, such as an HTML *Representation* and a .pdf *Representation*.

- The overall *Collection* may be composed of smaller *Collections*. Some of these may be static for the institution, such as the *ScienceCollection*, or determined dynamically, such as the *Collection* of all articles that contain .tiff3.0 *Files*. *Collections* may contain digital and non-digital objects.
- A *Journal* may belong to one or more *Collections*. It is the logical object describing all *Issues* with the same title (setting aside some complexities involving name changes, etc.).

- An *Issue* is part of a *Journal*. It is the logical object containing all of the *Articles* published in a single *Issue*.
- An *Article* is part of an *Issue*. It is the abstract concept representing a distinct intellectual creation – the article.
- An *Expression* is the specific intellectual or artistic form that an *Article* takes when it is realised. There may be multiple *Expressions* of an *Article* and each *Expression* may have multiple *Representations*.
- A *Representation* is a set of *Bitstreams* that are required to render the *Expression*. There might be several *Representations* of an *Expression* of an *Article* (e.g., an .html, a .pdf, an .xml, and a publisher-specific format).
- A *Bitstream*, such as a *File*, is part of a *Representation* (e.g., an .mp4 video File is part of an .html *Representation* of an *Article*).

This model supports an essential property of preservation activity. Institutions need to take a global view of their collections and resources in order to coordinate preservation activity and take the appropriate actions. It is not enough to consider each digital object in isolation. This is the reason that the model goes well beyond the individual digital object.

There are also smaller *Components* of an *Article*, such as a *TextStringComponent* or a *TitleComponent* which in themselves are *IntellectualEntities*. They can have several *Representations* with possibly (slightly) different *Characteristics* of their own, such as their *fontSize Values*. Each *Representation* is captured in one or more *RepresentationBitstreams*.

**Modelling Requirement 1.1.4:**

Management (or curatorial decision) determines the choice of *IntellectualEntity* instances.

Example: Library Catalogue

An organisation may choose to just model *Books* as *IntellectualEntities* without capturing information about *Representations* or *Bitstreams*. This is the case for a traditional universal bibliographic catalogue.

Example: Books in a Digital Repository

In an hypothetical organisation, *Books* are scanned as one .pdf file per page. *Books* are to be explicitly modelled in order to record descriptive information about them. The *Book*'s *Pages*, are *IntellectualEntities* in their own right. A page is the result of a decision to break up some text - either by an author, artist, typesetter or text editing programme. In the latter case the choice is made (maybe as low priority) by choosing to use that program and by choosing to create a book rather than, say, a scroll. This is a cultural expression. If this is considered an insignificant intellectual and curatorial choice the organisation does not model the corresponding *Page IntellectualEntity* explicitly. Instead it implements a *Book Representation* object as the set of all .pdf *Files*. This decision results in the modelling of one *Book IntellectualEntity*, one *Book Representation* and *Files*. These are the three levels on which the organisation chooses to manage the *Collection*. Other entities, such as the *Page IntellectualEntities* and *Page Representations* exist but are not modelled explicitly. If *Pages* were significant, such as in the "Turning the Pages" software (Armadillo Systems, nd) the organisation would choose to model it.

Most physical entities in the cultural heritage sector, such as a *Page*, are the result of an intellectual decision and therefore they have corresponding *IntellectualEntities*. But one is not always interested in managing them on this level.

**Modelling Requirement 1.1.5:**

When an organisation implements the conceptual model, it chooses which entities it models explicitly, and which it leaves implicit.

Example: "Turning the Pages"

For example in the "Turning the Pages" application (Armadillo Systems, nd) pages are annotated and managed on a page level. If one does not manage at this level one would not need to create an *IntellectualEntity* for it. One can still manage corresponding *Representations* or just corresponding *Bitstreams* for them, with the implicit understanding that an *IntellectualEntity* exists for every *Representation* or *Bitstream*. There is no inherent base case for *IntellectualEntities* that would represent the lowest possible granularity. It is a curatorial decision how finely one wants to model and that, in turn, is driven by business requirements and curatorial decisions.

**Modelling Requirement 1.1.6:**

- Every *Bitstream* has a corresponding *RepresentationBitstream* that it implements.
- Every *RepresentationBitstream* has at least one corresponding *Bitstream* that implements it.
- Every *RepresentationBitstream* has a corresponding *Representation* which contains it.
- Every *Representation* contains at least one *RepresentationBitstream*.
- Every *Representation* has a corresponding *IntellectualEntity*.
- Every *IntellectualEntity* has at least one *Representation* that implements it.

Since it is open to the user of the model whether to model all aspects of the system explicitly, the implementation may capture only some of these entities explicitly (even if they necessarily exist).

**Modelling Requirement 1.1.7:**

An *IntellectualEntity* can have several different *Representations*. If, however, for a given application, there is exactly one *Representation* corresponding to an *IntellectualEntity* it is the implementer's choice whether to model it as *Representation* or as *IntellectualEntity* or as both.

**Modelling Requirement 1.1.8:**

*RepresentationBitstreams* and *Bitstreams* model physical digital objects.

In a mixed repository of traditional and digital objects the actual physical items would be modelled on the corresponding level of *RepresentationBitstreams* and *Bitstreams*.

**Modelling Requirement 1.1.9:**

*IntellectualEntities* are not defined by whether they are used by humans or automatically. Types of use change over time and are not predictable. Examples of *IntellectualEntities* for machine use are the text segments defined in XCL (Thaller et al., 2008; Thaller, 2009) in order to describe the *SignificanceConstraints* on these text segments that need to be satisfied during *PreservationActions*. This is evaluated and quality assured by automatic systems.

**Modelling Requirement 1.1.10:**

An *IntellectualEntity* is not defined by the way its *Representations* are created. For example, a *Page* as an *IntellectualEntity* (see Modelling Requirement 1.1.3) can have several *Representations*, such as the OCR text and a .pdf. They were created for different purposes and in different ways, but describe the same *IntellectualEntity*.

**Modelling Requirement 1.1.11:**

Properties can be applicable to objects in every tier. For example:

- *observedFileSize*, *encoding* or *observedChecksum* are applicable to *Files*.
- *targetFileSize*, *targetEncoding* or *targetChecksum* are applicable to *RepresentationBitstreams*.
- *numberOfFilesInTheRepresentation*, *totalRepresentationSize*, *resolution*, or *preservationLevel* are applicable to *Representations*.
- *pageCount* or *frameRate* are *Properties* applicable to *IntellectualEntities* such as a journal article or video. *Alignment* is a *Property* applicable to a *TextComponent*. *SemanticInterpretation* can be a *Property* of any *Component IntellectualEntity*.

Vocabulary for PreservationObject sub-classes

- Extensible vocabulary including *IntellectualEntity*, *Representation*, *RepresentationBitstream* and *Bitstream* is discussed in their respective subsections below.
- They can be further categorised as illustrated earlier in this section.
- An orthogonal categorisation of *PreservationObjects* could be, for example, the intellectual content, the semantic and syntactic interpretation which are necessary to interpret the content, the format in which the content is encoded, or the physical realisation of the content.

### *3.1.2.1 Intellectual entities*

Definition of IntellectualEntity

A set of content that is considered a single intellectual unit for purposes of management and description; a distinct intellectual or artistic creation that is considered relevant, by curatorial decision, to a Designated Community in the digital preservation context.

Vocabulary for IntellectualEntity sub-classes

- Extensible vocabulary that conceptually describes an application area's functional decomposition can, for example, be found in the archival world in the form of a *Fonds*, *SubFonds*, *Series*, *SubSeries*, *Files* and *Items* hierarchies, or in the library world in the FRBR (FRBR, 1998) decomposition into *Work*, *Expression*; *Manifestation*, and *Item* or in *Collection* and *SubCollection*, hierarchies.
- Extensible vocabulary for *Component* sub-classes (such as *Header*, *Body*, *Footer* / *Title*, *Abstract*, *Appendix* / *SubString*, *Table*) is being developed in preservation characterisation research. For text-based systems the vocabulary to specify the *Component* sub-classes can, for example, be taken from the NLM DTD (NCBI & NLM, 2012; NLM, nd; NISO, 2012) or TEI (nd) which uses tags for mark-up of text *Components*. Other *Component* sub-classes can be defined for other content-type specific needs, such as sound, video, etc.
- The *Component* entity can be decomposed in several ways, such as
    - by the type of content (e.g., *TextComponent*, *ImageComponent*, *Table-Component*), or
    - by document structure (e.g., *HeaderComponent* or *TableOfContentComponent*).

An example is shown in Figure 22.

- *Values* for *Characteristics* of *IntellectualEntities* are mostly determined by humans, since, by their nature, they capture curatorial values. Some can be measured from their associated *Representations' Bitstreams*. For example, the *font* of a *CharacterComponent* can be extracted from its *Bitstream*.

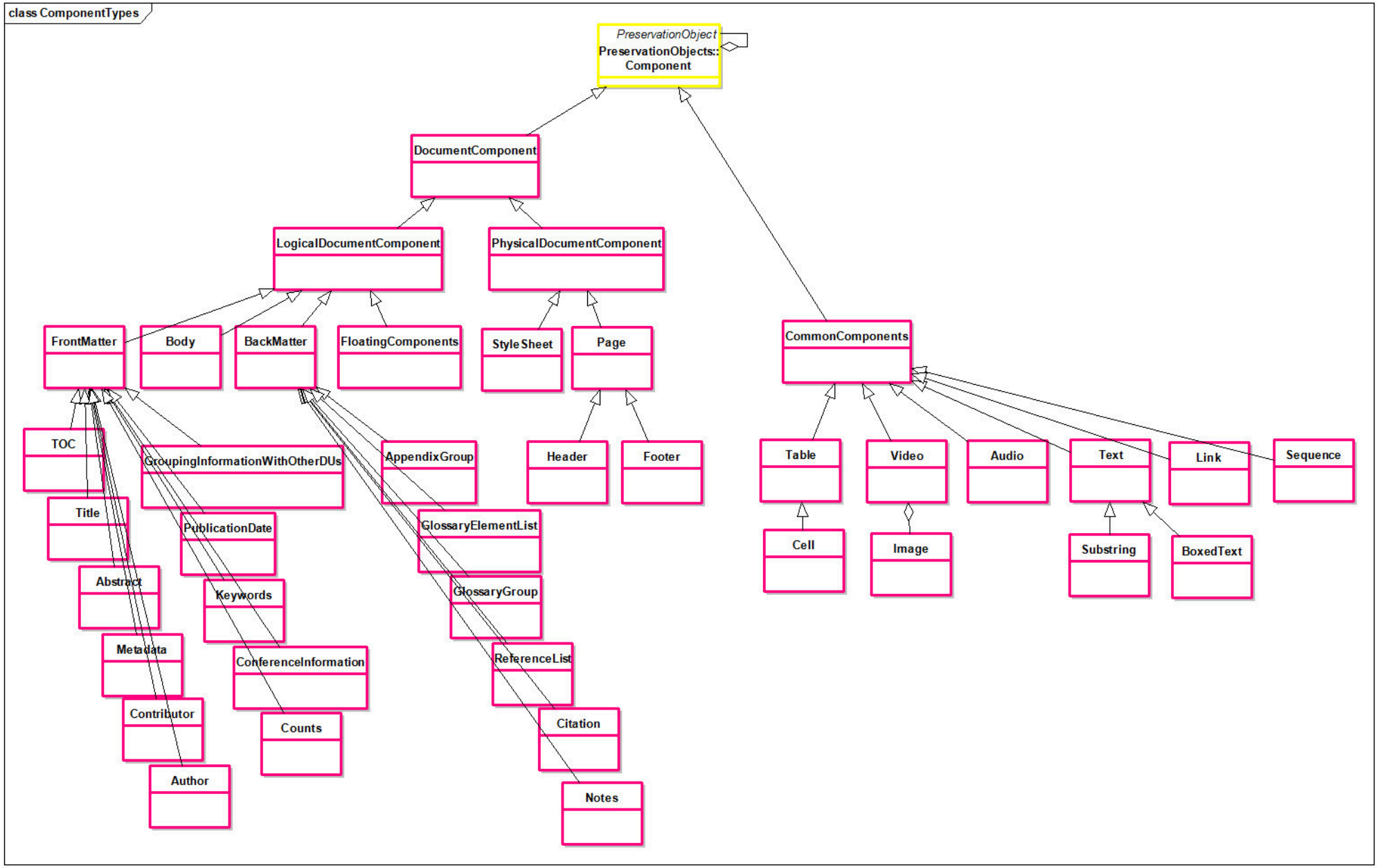



**Figure 22: Example of *Component* sub-classes for document applications based on the NLM DTD (NLM, nd)**

### 3.1.2.2 Representations

Definition of Representation

One physical embodiment of an *IntellectualEntity*.

The set of all *RepresentationBitstreams* that are needed to create one rendition of an *IntellectualEntity* together with the necessary structural information.

Modelling Requirements for Representation

**Modelling Requirement 1.1.12:**

*IntellectualEntities* may have multiple *Representations*. For example a journal article may come both in .doc format and an .xml document with associated *Files*. Any set of *Files* that allows authentic rendering of the *IntellectualEntity* within its technical *Environment* is a *Representation* of the *IntellectualEntity*.

### 3.1.2.3 Representation bitstreams

Definition of RepresentationBitstream

A *Representation* is a composition[16] of all the *RepresentationBitstreams* that are needed to create a single rendition of the *IntellectualEntity* that is embodied by the *Representation*.

*RepresentationBitstreams* are the logical, ideal bitstreams that make up the *Representation*. They may have *Properties*, such as their *targetFileChecksum* and *fileName* and *SignificanceConstraints* that need to be satisfied during *PreservationActions*. In contrast, *Bitstreams* are the physical, actual counterparts of *RepresentationBitstreams*. They, for example, may be corrupted and have *observedFileChecksums* varying from those of their *RepresentationBitstreams*. A *RepresentationBitstream* may be implemented by several *Bitstreams* if there is replication present in the system.

The term *RepresentationBitstreams* is used generically and may be implemented through specific sub-classes to include *RepresentationBytestreams* of various byte lengths or *RepresentationFiles*.

[16] In the UML sense

The bits representing *Characteristics* of *IntellectualEntities* may not necessarily align with byte boundaries, for example when they are extracted from a compressed *File* directly or if *Characteristics* are represented as bitmaps. They may span several *Files*, for example large *File*s may be split with a UNIX "split" command, data may be streamed into containers of a fixed file size, such as .arc, or data may be split over several *Files* to optimise access.

### *3.1.2.4 Bitstreams*

Definition of Bitstream

A *Bitstream* is contiguous or non-contiguous data within one or more *Files* that has meaningful common properties for preservation purposes.

It can be a digital *File*, embedded within a digital *File*, or spread across *Files*.

A non-*File Bitstream* can be transformed into a standalone *File* with the addition of *File* structure (headers, etc.) and/or reformatting the *Bitstream* to comply with some particular file format, by giving it the required metadata (name, create date, ...), a path, and placing it into a file system.

A *File* is a named and ordered sequence of bytes that is known by an operating system. A *File* can be zero or more bytes and has a file format, access permissions, and file system characteristics such as size and last modification date (PREMIS (2012)).

Vocabulary for Bitstream sub-classes

- *Bytestream* is a sub-class of *Bitstream*.
- *File* is a sub-class of *Bytestream*.

An example extensible vocabulary is shown in Figure 23.

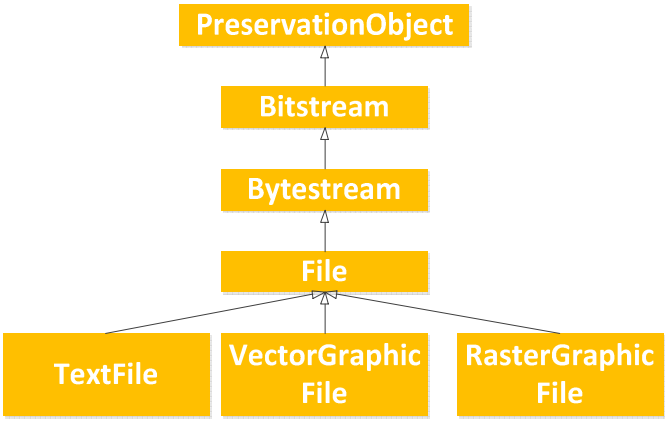



**Figure 23: Example of some *Bitstream* sub-classes**

### 3.1.3 The preservation object's environment

*PreservationObjects* do not exist in isolation. A user or system interacts with an object in an environment. Therefore, every *PreservationObject* is associated with one or more *Environments* that support different purposes or functions.

There is a close relationship between an *Environment* and an extended notion of Representation Information as it is defined in OAIS (CCSDS, 2012) since it is necessary to understand the *Characteristics* of *Environments* in order to understand the *PreservationObject*. Other examples of extended notions of Representation Information are discussed by Brown (Brown, 2008).

| Definition of Environment |
|---|

A set of factors which constrain a *PreservationObject* or *PreservationAction* and that are necessary to interpret it.

| Modelling Requirements for Environment |
|---|

**Modelling Requirement 1.2.1:**

Every *PreservationObject* has one or more *Environments* which may be fulfilling different purposes and functions.

Examples of *Environment* purposes include delivery (remote or local), creation, ingest, and preservation. For example, a *File* or a *Representation* object may have creation, ingest, preservation, and access *Environments*; a *Collection* may have an internal, a physical delivery, and an online delivery *Environment*.

Examples of *Environment* functions include rendering, editing, executing, and printing.

**Modelling Requirement 1.2.2:**

*Environments* have *Characteristics*. For example:

- *memoryUsage* = "low" is a *Characteristic* of a software tool *Environment* that renders the *PreservationObject*.

**Modelling Requirement 1.2.3:**

The selection of a *TransformationPreservationAction* may depend on the *Characteristics* of *Environments* and the *Characteristics* which the output

*Environment* would have if the given candidate *TransformationPreservationAction* was to be executed. For example, if the output *fileFormat* of a migration is not supported in the organisation's *SoftwareEnvironment* then this type of migration is not an attractive *PreservationAction*.

**Modelling Requirement 1.2.4:**

It may not be possible to derive the output *Environment* of a *Transformation-PreservationAction* from a *File*'s input *fileFormat* alone. For example, if a *File* does not make use of the full range of features of its *fileFormat* then it can be supported by an *Environment*, which in general would not support all *Files* of this *fileFormat*. If, for example, a .doc *File* contains only text without formatting, headers and tables, and so forth, then a .txt output might be considered perfectly adequate, even though this would in general not be considered an ideal migration format for a .doc *File*.

**Modelling Requirement 1.2.5:**

Stakeholders may wish to specify their intentions (necessary, recommended, acceptable…) together with the *Environment*, as is recommended in the PREMIS data dictionary (2012).

**Modelling Requirement 1.2.6:**

An *Environment* that is a *PreservationTool* can take the role of an *Agent*. If the *Agent* is used it performs a *PreservationService*. See section 3.2.2 for the definitions. For example:

- *numberOfIntermediateCopies* <= 3 and *preservesColourDepth* = "yes" are *Characteristics* of an *Environment* that takes the role of an *Agent*. They can be captured in a *Tool* or *Environment* registry.

**Modelling Requirement 1.2.7:**

An *Environment* can take the role of a *PreservationObject*. In this case the *Environment* itself is the target of preservation. For example:

- A computer game is to be preserved. This requires the preservation of the whole gaming *Environment* including the software and all the information needed to migrate (port) it, emulate it or virtualise it and later reuse it.

**Modelling Requirement 1.2.8:**

*Environments* are related to other *Environments* in a variety of relationships. In particular, they can form a representation information network. The 'Preserving Virtual Worlds' project has enumerated some of them for virtual worlds (McDonough et al., 2010). Software package dependency types in the Debian (Debian, nd) system is another list of *Environment* relationships.

**Modelling Requirement 1.2.9:**

*Environments* for *PreservationObjects* at a higher level (logical or representation layer, resp.) also apply to *PreservationObjects* at a lower level (representation or physical layer, resp.).

The *Environment* for a *File*, for example, can be different from the *Environment* of the *Representation* to which it belongs. As long as the *File* is part of its *Representation*, it will live in the *Representation's Environment*. When it is taken out of the *Representation*'s *Environment*, for example to be used in a migration, then the *File*'s individual *Environment* will influence the *Environment* of its new *Representation*. For example, a website may only render properly in IE6.0, but a .jpg image contained within it would render in a simple viewing environment.

**Modelling Requirement 1.2.10:**

DePICT introduced the distinction between abstract and physical *PreservationObjects*, particularly between *IntellectualEntities* and *Bitstreams*. There is also a distinction between abstract, generic and concrete, physical *Environments*. Abstract *Environments* may, for example, be described in a registry. There are increasing levels of granularity in the description of abstract *Environments*. A concrete software *Environment*, for example, may take the form of a source code *File* and itself be subject to preservation.

Vocabulary for Environment sub-classes

Every *Environment* may be broken down into sub-*Environments* that are needed for the interpretation and representation of a *PreservationObject*.

Examples include hardware and software environments, the community, budgetary factors, the legal system, and other internal and external factors of political, economic, social or sociological, technical, legal or legislative, or environmental nature.

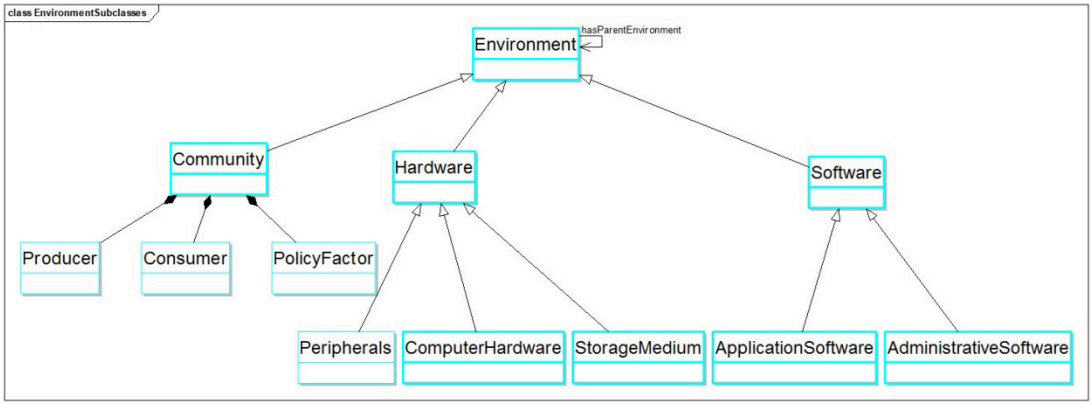



**Figure 24: Top-level vocabulary for *Environment* sub-classes**

Example top-level vocabulary to specify the *Environment* sub-classes is illustrated in Figure 24. Lower-level vocabulary is specified in Figure 25, Figure 26 and Figure 27. The *Environment* entity in this conceptual model is extensible according to institution-type specific needs. There are on-going efforts to model *Environments* for digital preservation purposes in detail. Examples are the JISC- funded "The Significant Properties of Software: A Study" (Matthews et al., 2008) and SWOP projects (SWO, nd; SWOP, nd), the TOTEM model in the KEEP project (Delve, 2011; Delve, Anderson, 2012; TOTEM, nd), and the context definition workpackage in the TIMBUS project (Edelstein et al., 2011; TIMBUS, nd). The *Environment* entity can be refined from detail from these efforts.

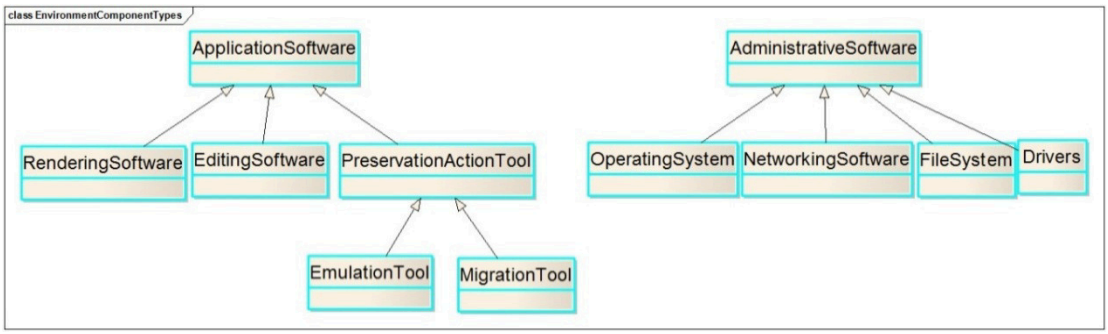



**Figure 25: Vocabulary for Software**

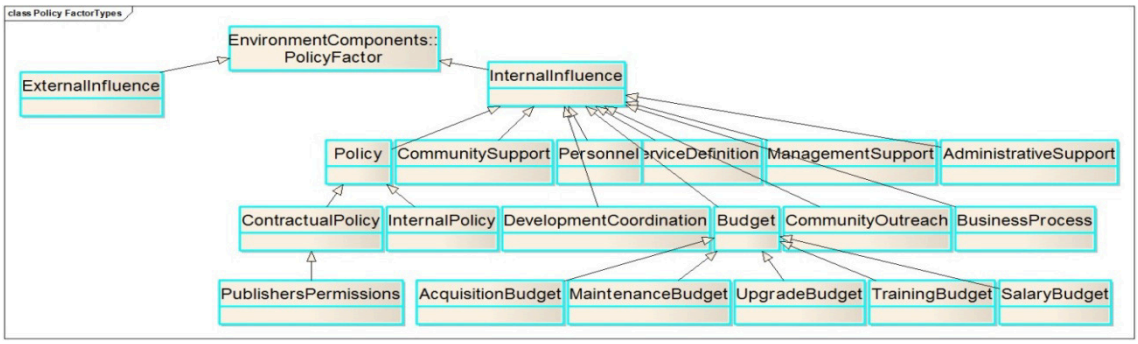



**Figure 26: Vocabulary for internal influences**

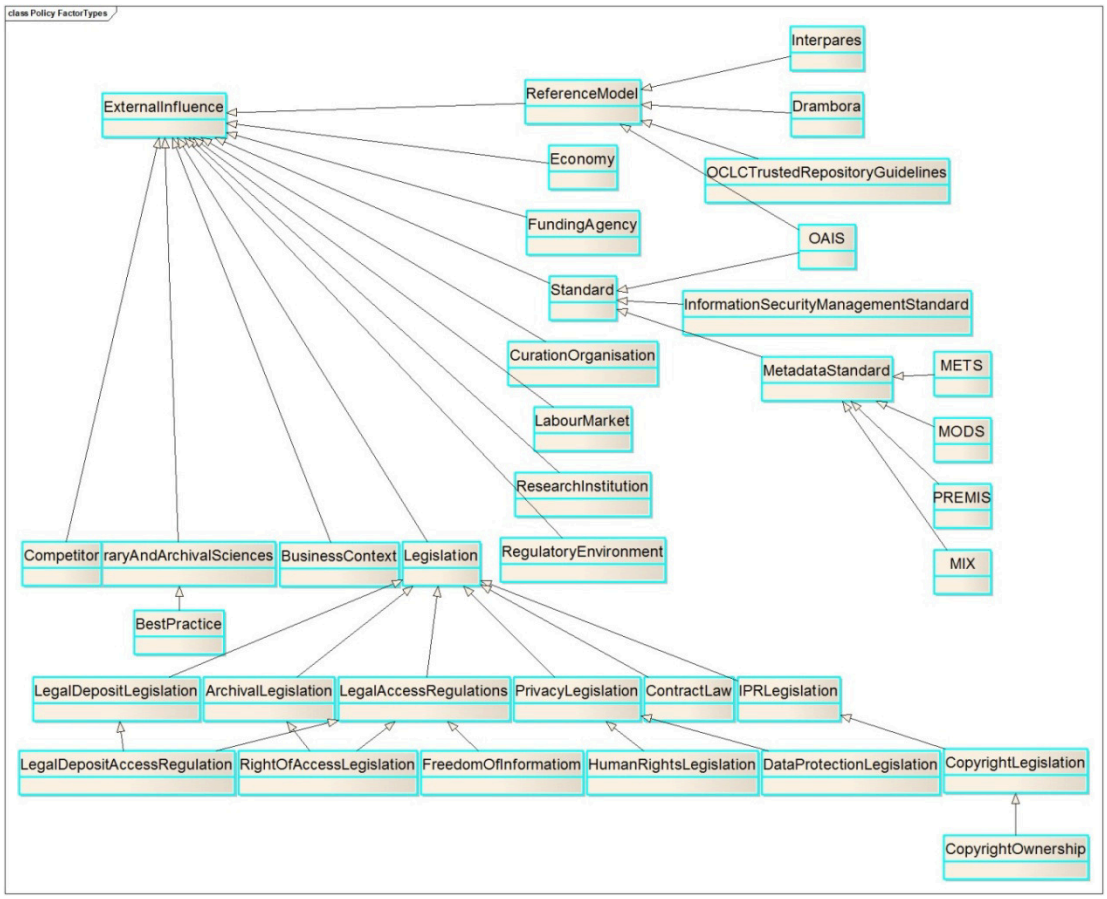



**Figure 27: Vocabulary for external influences**

### 3.1.4 Properties and characteristics

In order to meaningfully reason about entities, such as *PreservationObjects* and *Environments* it is necessary to know their *Properties* and *Characteristics*. DePICT defines them as follows

- Entity – Anything whatsoever.
- Class – A class is a set of entities. Each of the entities in a class is said to be an instance of the class.
- *Property* – A *Property* is an entity that is not a class that names a relationship.
- *Characteristic* – A *Property* / *Value* pair associated with an entity. The *Value* is an entity. This relationship was illustrated in Figure 6.

This section discusses the specific *Properties* of the digital preservation domain and the requirements for *Properties* and *Characteristics*. It illustrates these *Properties* and their requirements with specific examples, but makes no effort to compile a dictionary or ontology of *Properties* of, for example, files, software tools, or file formats. There are separate efforts in the digital preservation community to grow the vocabulary for specific *Properties*, such as in PREMIS (PREMIS, 2012), and PRONOM (Brown, 2005; TNA, nd).

#### 3.1.4.1 Properties

Definition of Property

An abstract attribute, trait or peculiarity suitable for describing *PreservationObjects*, *PreservationActions* or *Environments*.

The model's scope is limited to *Properties* which are observed to be used and to be useful for achieving digital preservation goals.

Modelling Requirements for Property

**Modelling Requirement 1.3.1:**

A *Property* applies to a class if it can be meaningfully associated with some instances of the class. *Properties* are applicable to *PreservationObject*, *Environment* or *PreservationAction* sub-classes.

**Modelling Requirement 1.3.1a:**

Many *Properties* are applicable to specific subsets of *PreservationObjects*. For example, the *Property fontSize* is applicable to *TextComponent PreservationObjects*; it would not be applicable to an *AudioComponent PreservationObject*.[17]

**Modelling Requirement 1.3.1b:**

This model associates a *Property* with the type of *PreservationObject* to which it applies (see *appliesTo* in the section 7.1.4), rather than with a file format. Examples include *Bitstream*, *Representation*, *IntellectualEntity* (e.g. *e-Book*, *SoundRecording*, *TextComponent*, *TableOfContents*. A *Property* applies to a file format only if it characterises the format itself, rather than an object encoded in it.[18]

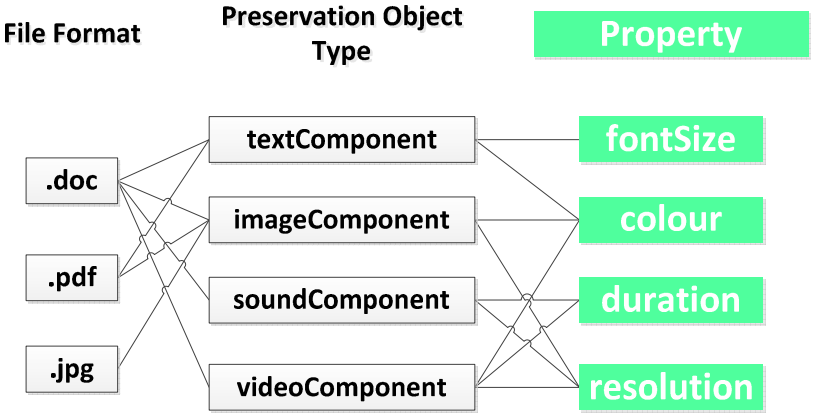



**Figure 28: Mapping of applicable *Properties* to *fileFormats* via *PreservationObject* type**

For example, a *Component*, such as a *TextComponent* or *ImageComponent*, is associated with a set of applicable *Properties*. If this *Component* is implemented by a basic file format[19] then the applicable *Properties* apply by transitivity to the file format. See Figure 28 for an illustration. This approach makes it explicit that the *fontSize Property* applies to the .doc file format because it applies to *TextComponent* objects.

[17] The association of *Properties* with types of *File* objects is discussed in the Planets Testbed (Aitken et al., 2008). The model generalizes this to the other types of *PreservationObjects*, especially to the *Bitstreams* that represent *IntellectualEntity Components*, such as text, sound and image *Components* contained in a *File*.

[18] File format *Properties* can, obviously, factor into preservation decisions if they, for example, determine the choice of file format. Examples of file format *Properties* are captured in the Library of Congress (nd-a) "Sustainability of Digital Formats" service.

[19] without embedded, dependent or linked objects in other file formats

A *Property* also applies to *PreservationAction*, or *Environment* (e.g. *Legal-Environment*, *OperatingSystem)*.

**Modelling Requirement 1.3.2:**

The language that the model uses to define *Properties* must be expressive enough to have a *Property* refer to a combination of *PreservationObjects*, *Environments*, or *PreservationActions*. Consider the relative size of two images, the absolute distance of a line from the text, the metrics describing column layout. These all refer to several objects. This means that Modelling Requirement 1.3.1 is generalised to say that a *Property applies* to a vector of classes if it can be meaningfully associated with some instances of the classes – i.e. *Properties* can be n-ary.

**Modelling Requirement 1.3.3:**

Every *Property* applies to exactly one vector of *PreservationObject*, *Environment*, or *PreservationAction* sub-classes. A *Property* with the same name can be defined for other vectors of Classes, but should have a different globally unique *PropertyIdentifier*.

**Modelling Requirement 1.3.4:**

*Properties* are related to each other and their relationships have to be modelled explicitly.

**Modelling Requirement 1.3.4a:**

In many cases, it is useful to define one *Property* in terms of others. For example, the *aspect-Ratio* of an image might be defined as *imageWidth* / *imageHeight*. For example *duration* can be calculated from *dateTimeRange*. As a result, it is essential to record how such *Properties* are defined and derived in order to ensure consistency.

**Modelling Requirement 1.3.4b:**

Some *Properties* are modelled hierarchically. For example *maintenanceSalaryCost* is a kind of *maintenanceCost* which is a kind of *budgetCost*. Furthermore, different file formats have similar, but not identical *Properties*. A data model of *Properties* should be able to capture the relationships between them and specify how to compare or convert them. Figure 29 illustrates this.

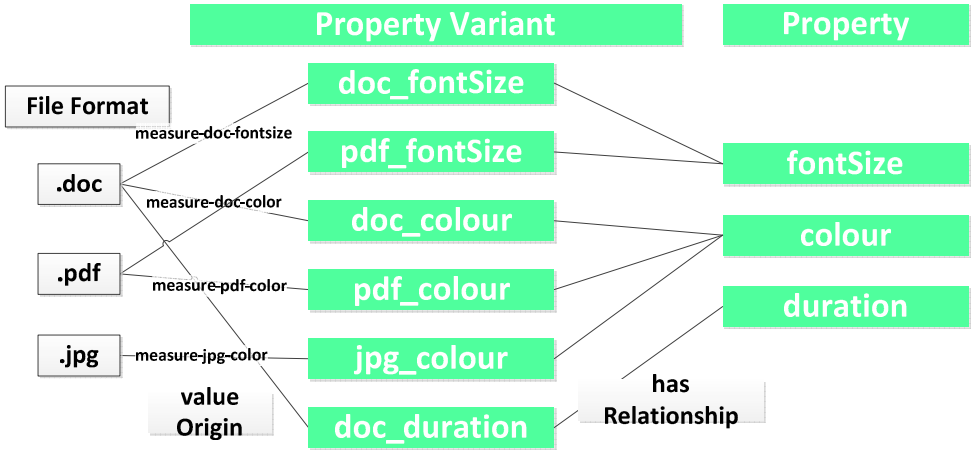



**Figure 29: *ValueOrigins* and relationships between *Properties***

**Modelling Requirement 1.3.4c:**

Relationships between *Properties* are sometimes difficult to capture in digital preservation tasks. A key task of many *PreservationServices* is to compare *Values* of *Properties* of a digital object before and after a *PreservationAction*, such as a migration, in order to assess the quality of the *PreservationAction*. This may be hard to do if the *Properties* that are applicable to the file formats are incompatible. This section discusses the reasons for this.

Some related *Properties* are hard to compare across file formats because those formats are represented in fundamentally different paradigms with different **primary components** and **content structures**.

**Properties for file format paradigms with different primary components**

Each file format has primary components. *Properties* apply to those components and are used to characterise a digital object of this file format. For example, a *SubstringComponent* of a text document has *Characters* as the primary component that can be described by the *fontType*, *fontColour*, and *fontSize* properties. When file format paradigms use different types of primary components, *Properties* may not be easy to compare.

For example, both a Word document and a .pdf document may represent the same text, but their underlying paradigms are quite different. .pdf documents' primary components are **representation elements**, such as elements of the page layout. Their *Properties* describe a fixed-layout 2D document with an underlying page orientation. Word documents' primary components are **content elements**, such as *TextStrings*, *Columns*, or *Titles*. Their *Properties* describe them mostly independent of the page layout; for example, Microsoft Word

has no notion of the *PageCoordinatePoints* where a *Paragraph* starts. This results in a phenomenon where seemingly identical *Properties* can actually refer to quite different *Properties*. For example, the *Property pageNumber* in Microsoft Word is determined by the author of the document. It may start with page numbering of a title page, or start after an introduction to the document. The .pdf document displays *pageNumbers* starting with the first physical page. Even though it may display a different logical *pageNumber*, it has no "awareness" of it.

Likewise, both vector graphics and raster graphics capture images. But while vector graphics describe the properties of content elements of the image (such as the *Width*, *Length* and *Colour* of a *Line*, or the *Diameter* and *Position* of a *Circle*), a raster image would represent the same content by recording *Properties* of its representation elements, the *Pixels* of the image. Raster image formats have no notion of *Properties* of *Lines* and *Angles*; vector graphics formats have no notion of *Pixel Properties*.

**Properties for file format paradigms with different content structures**

Even though both the Open Document Format for Office Applications (ODF) and Office Open XML (OOXML) have content elements as primary components, their *Properties* are not necessarily directly comparable because they use different models of how the text is structured. ODF uses a hierarchical content element decomposition into *Chapter*, *Section*, *Paragraph*, *MarkedUpText*, etc. *Properties* apply to those structures. OOXML, however, applies its *Properties* to runs of consistent mark-up which can span structural elements, for example, mark text as bold across paragraphs. In this case, one needs to not only capture the relationship between the *Properties*, but also the relationship of the clashing structural elements.

**Properties describing absolute and relative page layout**

In addition to differing primary components, file formats fundamentally differ by whether they have absolute vs. relative page layout. Of the example formats in this section, the image and .pdf formats describe the absolute position of their content or representation elements, while Word and ODF documents describe the relative position of their content elements. Any *Properties* describing positions on a page or positions of components relative to each other are hard to capture in their non-native representations.

**Different scope of functionality of file formats**

Different file formats support different functionality. For example, OOXML has editing sessions, for which it records a modification and editing history. This functionality is not supported by some other file formats. It is therefore hard to compare *Properties* relating to this differing functionality across file formats.

**Relating properties for different file format paradigms**

In order to compare *Properties* across content types, such as image or text, it is necessary to explicitly establish relationships between their different *Properties*. *Font Properties*, for example, may cross text and image paradigms. Properties of fonts that are encoded as images cannot be easily compared to those of fonts that are encoded as characters. In the example in Figure 30, in order to compare *fontColour* before and after migration, it is necessary to establish that the *fontColour* of the *Letter* is the same as the *pixelColour* of every *Pixel* in the corresponding raster image.

**Figure 30: Example. *Font Properties* are difficult to establish and compare across content types (depicted: text representation and raster image representation)**

Even if one remains within one content type 'image', if one works within the paradigm of raster images, then pixel properties are easily extractable. From this perspective comparing *Properties* to vector graphic elements can, at best, be heuristically approximated. If one works within the paradigm of vector images, then graphic elements are the primary components with measurable *Properties*. From this perspective, raster image pixel properties are not measurable and need to be related heuristically.

Due to the inherent conceptual distance, shifting from one file format paradigm to another, results in inaccuracies which make a reliable comparison based on *Properties* hard. For example, one can convert a vector graphic into a raster image in order to compare it with another raster image to infer their similarities or differences. But the conversion algorithm does not necessarily produce a raster image that has pixel-wise equivalence to another raster image of the

same content. This means that comparison metrics need to be developed that can anticipate the resulting inaccuracies while still capturing actual content differences.

This discussion is supposed to illustrate that in the preservation process interesting relationships between digital object *Properties* occur that are not straight-forward to resolve. A *Property* ontology is a way of modelling them explicitly in order to overcome possible misalignments. It illustrates that there is a need to capture the semantics of similar *Properties*. It illustrates why there is a need to define derived *Properties*. It also illustrates why there is a need for robust aggregate comparisons of digital object *Property Values*.

Example for Property

The example in Figure 31 illustrates how a *Property* may be expressed solely in terms of model elements and vocabulary. It is the definition of the *Property imageSizeWidth* for an *ImageFile PreservationObject*. All elements and vocabulary are taken from the model's conceptual detail definition in appendix 7.1.

This *Property* definition has 3 types of *Units*: inches, centimetres, and points. They all are valid alternative *Units*, however, if not specified, it is assumed that "points" are the default *Unit*.

The *Value* may be assigned and stored as metadata by digitisation software DigitizR on creation of the image. Alternatively *imageSizeWidth* may be derived in three ways. It may be characterised from an existing *File* by the JHOVE file format characterisation tool. It can be calculated if *aspectRatio* and *imageHeight* are known by using the conversion function associated with these *Properties*. Alternatively it can be derived if the *Property imageSizeWidth*_GIF is known, since they are known to be equivalent *Values*.

*Property*

| |
|---|
| *propertyIdentifier*<br>*http://ontology.xxx.yyy/1234* |
| *propertyName*<br>*imageSizeWidth* |
| *propertyDescription*<br>*The width of an image. No default value is provided. The default measurement unit is points.* |
| *appliesTo*<br>*ImageFile* |
| *range*<br>• *hasUnit*<br>*<UnitID for inches>*<br>• *dataConstraint*<br>*positive or zero float* |
| *range*<br>• *hasUnit*<br>*<UnitID for centimeters>*<br>• *dataConstraint*<br>*positive or zero float* |
| *range (isDefault="yes")*<br>• *hasUnit*<br>*<UnitID for points>*<br>• *dataConstraint*<br>*positive or zero int* |
| *hasDefaultValue*<br>*n/a* |

| |
|---|
| *hasValueOrigin*<br>• *hasValueOriginID*<br>*<ValueOriginID for JHOVE Version 1.1 extractor of imageWidth>*<br>• *isDefault*<br>*"no"* |
| *hasValueOrigin*<br>• *hasValueOriginID*<br>*<ValueOriginID for DigitizR Version2.5 - imageWidth>*<br>• *isDefault*<br>*"no"* |
| *hasValueOrigin*<br>• *hasValueOriginID*<br>*<ValueOriginID for conversion from aspectRatio and imageSizeHeight>*<br>• *isDefault*<br>*"no"* |
| *hasValueOrigin*<br>• *hasValueOriginID*<br>*<ValueOriginID for conversion from gif_format_imageWidth>*<br>• *isDefault*<br>*"no"* |
| *hasEvent*<br>*<CreationEventID giving creation time and author of this property>* |

*ValueOrigin*

| |
|---|
| *valueOriginIdentifier*<br>*<ValueOriginID for conversion from aspectRatio_imageSizeHeight>*<br>*valueOriginName*<br>*"conversion from aspectRatio and imageSizeHeight to imageSizeWidth"*<br>*hasTechnique*<br>*<conversion function> (aspectRatio(self), imageSizeHeight(self))* |

*ValueOrigin*

| |
|---|
| *valueOriginIdentifier*<br>*<ValueOriginID for conversion from gif_format_imageWidth>*<br>*valueOriginName*<br>*"conversion from gif_format_imageWidth to imageSizeWidth"*<br>*hasTechnique*<br>*"is same"* |

**Figure 31: Example *`Property`*'*`imageSizeWidth`*' and 2 different definitions of *`ValueOrigin`* for this *`Property`***

Vocabulary for specifying Properties

This thesis discusses the specific properties of *Properties* and *Characteristics* in the digital preservation domain. It illustrates these *Properties* with specific examples, but makes no effort to compile a dictionary or ontology of *Properties* of, for example, files, software tools, or file formats. The community goal is to have a deep vocabulary or ontology that would be generally acceptable and sharable by different stakeholders. There are separate efforts in the digital preservation community to grow the vocabulary for specific *Properties*. The appendix of Planets report (Dappert, Ballaux, Mayr, van Bussel, 2008) lists an initial collection of *Property* vocabulary for a subset of the *Environment* and *PreservationObject* sub-classes. The KEEP TOTEM (Delve, 2011; Delve, Anderson, 2012; TOTEM, nd) database captures the *Properties* of technical *Environments*. Planets created a *Property* ontology (Planets - XCL project, nd) for file formats. Metadata initiatives for descriptive metadata (MODS (nd), DC (nd), MARCXML (nd), TEI (nd), NLM (NCBI & NLM, 2012; NISO, 2012; NLM, nd), etc.), technical metadata (MIX, nd; TEXTMD, nd) and preservation metadata have elaborated useful vocabularies for *Properties*. For example, the PREMIS (2012) preservation metadata defines *Properties* for *Representations*, *Files* and *Bitstreams*.

### *3.1.4.2 Property values and their origins[20]*

Definition of ValueOrigin

The *ValueOrigin* specifies how a *Value* of a *Property* is obtained. It comprises the tool and algorithm used to determine a *Value* and the types of sources from which they can be obtained.

Modelling Requirements for ValueOrigin

**Modelling Requirement 1.3.5:**

*Values* for *Characteristics* may be stored or derived on demand. On-demand derivation can take place through characterisation services or through retrieval from registries or inventories[21]. Whether they are stored or derived needs to be recorded, since different *PreservationServices* will be chosen based on this *Property*.

**Modelling Requirement 1.3.6:**

This section suggests a categorisation of *ValueOrigins* that is based on studies of Planets project (Farquhar, Hockx-Yu, 2007; Planets, nd) systems. It shows

[20] The results reported in this section have been published in (Dappert, 2010).
[21] Such as software licenses, hardware inventories, standards and XML schemata in use, staff skills, etc.

- how the *Value* for the same *Property* can be obtained in different ways.
- how related *Properties* can have clashing, observed *Values*.
- how different *Properties* can be related to each other to derive one *Property's Value* from others. This can help to mitigate the *Property* clashes described in the previous section 3.1.4.1.
- Limitations of relating *Properties* to each other.

**Manually Assigned ValueOrigins**

**Category description:**

When *Values* are assigned manually they often need to comply with conventions, such as cataloguing rules, standards, controlled vocabularies, etc. This should be specified as part of the *ValueOrigin*.

**Derivability:**

Manually assigned *Values* can be created, stored and looked up or created on demand.

**Automatically Assigned ValueOrigins as a side-effect of a service**

**Category description:**

Regular internal operations, such as ingest, digitisation, and harvesting of digital objects, purchase of hardware and software, decommissioning of equipment, hiring, training and laying-off of staff, getting and spending money, or executing *PreservationActions*, all change *Characteristics* of *PreservationObjects* or their *Environments*. Equally, external operations, such as introducing a new *fileFormat* or a new *PreservationService*, change *Characteristics*. These *Value* changes need to be captured if they serve as a basis for making preservation decisions.

**Examples:**

- The *contentType* of *PreservationObjects* in an electronic journal ingest system is always set to "eJournal" upon ingest.

- The *budget* of an institution may be set during the execution of a *PreservationAction*: *preservationBudgetSize* := *preservationBudgetSize* – *preservationActionCost*.

**Derivability:**

Automatically assigned *Values* are created at times different from their use and have to be stored explicitly and looked up when needed.

**Extractable, File-Based ValueOrigins**

**Category description:**

The *ValueOrigin* is a function of the simple digital object: f(object).

The original source of *Values* may be a *File*, *Bytestream* or *Bitstream*. *Values* are extracted using a tool which implements an algorithm. The *ValueOrigin* should specify the algorithms and tools used. For effective, scalable preservation, the tool would support automatic extraction of *Properties* but this is not obligatory.

**Examples:**

- *imageWidth*
- *colourSpace* in .png and other formats
- *linkURLs* in HTML
- *numberOfAudioChannels*
- *bitstreamSize* may be extracted from the *Bitstream* object
- *MIMEtype* can be extracted using the JHOVE format characterisation tool or extracted from the *File* header.
- *colourFidelity* can be measured by *averageColour* or by *histogramShape*.
- *wordCount* can count hyphenated words as one or as multiple words.

**Derivability:**

Algorithms for *Value* extraction are based on file format specifications. This category is implemented for basic file-format-based *Properties* in preservation characterisation services, such as the XCL services or JHOVE (JSTOR, Harvard University Library, 2012).

**Extractable, Complex *ValueOrigins***

**Category description:**

The *ValueOrigin* is a function of a complex *PreservationObject* and/or the object's *Environment*:

f (*PreservationObject*$_1$, …, *PreservationObject*$_n$, *Environment*).

These are *Property Values* that cannot be taken from the *File* alone, but rather need to be extracted from

- a *Representation* – that is, the set of *Files* that makes up one complete rendition or execution of a digital object (such as an .html *File* with its embedded .jpg *Files*).
- a *Representation* including auxiliary *Files* (such as style sheets, non-embedded fonts, java scripts in .html *Files*, and schema definitions).
- the whole rendering stack (i.e. the *PreservationObject*'s processing and presentation software and hardware *Environment*).

These *Properties* are not captured in a file format specification alone but are based on the whole *Environment* as depicted in Figure 32.

.

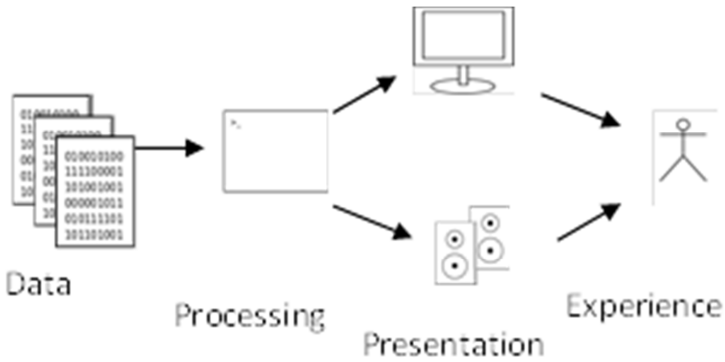



**Figure 32: Digital objects and their rendering stack. (Adapted with permission from Jan Schnasse)**

**Examples:**

- A Microsoft .doc document contains a link to a .jpg *File*. One needs to look at both *Files* to infer *Characteristics* about the image's appearance in the document.

- The *colour* of a hyperlink in an .html *File* is determined by the accompanying style sheet. Both *Files* need to be considered to characterise the *colour* of the hyperlinks.
- The presentation of an .html *File* depends on browser settings or the choice of browser. *Characteristics* vary depending on configuration.
- The actual layout of a Microsoft .doc document on paper depends on the printer driver.
- *imageWidth* can be obtained from the rendering software, e.g. Adobe Photoshop.
- *fileSize*, since it depends on the operating system, is derived by asking the file system, rather than counting the actual bytes.

Current characterisation tools are defined to work on *Representations*. Most often one characterises digital object *Representations*, but one can also characterise at a higher level, e.g. *Collection* profiling tools analyse *Characteristics* of a *Collection* at a given time and measure *Values* for *Properties* such as *fileSizeDistribution*, *fileFormat-Distribution*, *fileFormatFrequency*.

**Derivability:**

This is a generalisation of characterising one *File* at a time without regard to its *Environment*. Once one includes multiple *Files* and *Environments* into ones' scope, the set of automatically extractable *Properties* is expanded. This category could be implemented now. Some very useful information can be extracted easily; but some with, sometimes, considerable effort.

**Non-Extractable, Complex ValueOrigins**

**Category description:**

The *ValueOrigin* is a function that approximates the *Property*'s *Value*

f' (complex *PreservationObject*, *Environment*) ≈ f (complex *PreservationObject*, *Environment*).

These are *Properties* that are too complex to capture reliably in an algorithmic way, but they can be approximated by related metrics.

**Examples:**

- The stakeholders' observation of *imageQuality* does not always align with existing image quality metrics. But it is possible to define an acceptable metric which can be measured and compared (Heydegger, 2009).
- Different parameter configurations of *frequencies*, *amplitudes* and *modulations* can produce comparable sound to the human ear. Even if the *Representations* are not identical, they can have an identical effect for the user. In this case, the *Property perceivedSound* is an approximate metric which maps the measurable sound *Properties* onto it.
- Pixel-wise different images may have the same effect on the human eye or rendering devices, since some differences cannot be perceived or rendered.

Multiple metrics can be created to define which combinations are perceived as the same *imageQuality*, *sound* or *colour*, respectively.

**Derivability:**

By definition, these *Characteristics* cannot be inferred from extractable *Characteristics* unless an algorithmically supported metric is developed. This category can be implemented now, but with, sometimes, considerable effort for development of the algorithmically supported metrics.

**Implicit Semantics ValueOrigins**

**Category description:**

The *ValueOrigin* is a heuristic that results in a *Value*, as well as a confidence measure. The *Value* and confidence measure are repeatable and always give the same results.

(f ' (complex *PreservationObject*, *Environment*, heuristic),

conf (complex *PreservationObject*, *Environment*, heuristic))

These are *Properties* that require interpretation of semantics that is not captured in the *PreservationObject* and its *Environment*. This can, for example, be achieved by employing knowledge-based heuristics.

**Examples:**

- Some CAD drawings of pipes only specify where pipes are, but not how they are connected. The connections may be clear to the user, but difficult to extract from the *PreservationObject* and its *Environment*.
- Older .pdf formats do not have structural *Component IntellectualEntities* such as *Titles*, *Abstracts*, *and Footers*. Even in newer .pdf formats, functions supporting structural *Components* are currently rarely used in practice during the document creation process. They can, therefore, not be reliably automatically identified.

**Derivability:**

Implicit semantics require knowledge-based reasoning to infer *Property Values*. The *Property Values* in this category can be determined reliably and repeatably, but with considerable effort.

**Inferable ValueOrigins**

**Category description:**

The *ValueOrigin* is a composite function of other *ValueOrigins*:

f($g_1$ (*PreservationObject*), …, $g_n$ (*PreservationObject*)).

These are *Properties* that are not explicitly captured in the file format, but can be inferred from other *Properties*. *Values* may be inherited in a *PreservationObject* or *Property* hierarchy, derived through a function from other *Values*, or logically inferred.

The *ValueOrigin* should specify the algorithm that can be used to infer it.

This can also be used to relate *Properties* that have synonymous names, by explicitly stating their equivalence.

**Examples:**

- *aspectRatio* of an image may be calculated as *imageWidth* / *imageHeight*.
- *colourFidelity* can be measured from either of two different functions: *averageColour* or *histogramShape*.

- *wordCount* can be measured in several ways: e.g., count hyphenated words as one or as multiple words.
- *resolutionInPPI* can be mapped via its data type to *resolutionInLinesPerMillimeter.*
- *imageWidth* of an image, used as *Property* in one file format, may be inferred from the *Property width*, used in another file format, by stating its equivalence with *width*.
- *bitDepth*, is described as one non-negative number in .png and as three non-negative numbers (one per colour channel) in .tiff. Even though the *Property* is the same in both cases, they have different data types for their *Values*. This can in many cases be expressed through a functional relationship with which one can be derived from the other.
- Sometimes only a subset of *Values* can be inferred. Using an ICC colour profile one can for example, infer some CMYK colour *Values* from given RGB *Values*. But the two colour spaces don't overlap completely and there will be no equivalent *Value* for every given *Value*.

**Derivability:**

Algorithms for the *Value* inference need to be defined. Even though this category can be implemented now, it has not widely been done. The *Property Values* in this category can be determined reliably and repeatably.

The specification of how the involved *Properties* are related can be used to resolve clashes in levels of granularity between *PreservationServices* as discussed in section 3.1.4.1.

**Non-Predictable ValueOrigins**

**Category description:**

The *Property Value* is always the same, but the observed *Value* can be different at different times, for example due to interpretation.

f (complex *PreservationObject*, *Environment*, interpretation)

These are *Characteristics* that possibly have different observed *Values* when evaluated by different mechanisms (e.g. different people or the same person at different times).

**Examples:**

- *colourVibrance* can be judged differently by different observers.

**Derivability:**

The *Property Values* in this category can, by definition, not be reliably inferred.

For testbed purposes, the statistical average of these *Properties* may well be determinable. For example, the Mean Opinion Score metric (Reckwerdt, nd; ITU Radiocommunication Assembly, 2002) may be used for this purpose. But for the individual digital object, these techniques cannot be applied.

**Time Varying ValueOrigins**

**Category description:**

The *Property Value* is different at different times, depending on environmental changes. The observed *Value*, therefore, can be different at different times.

f (complex *PreservationObject*, *Environment*, time)

These are *Properties* whose *Characteristics* cannot be reliably reproduced because of time varying behaviour or *Value* change over time.

**Examples:**

- A time varying sequence of images in an .html table cell, such as flashing advertisements, will result in different extracted images at different times.

**Derivability:**

The *Property Values* in this category can, by definition, not necessarily be repeatably inferred.

**Indeterminable ValueOrigins**

**Category description:**

The *Value* cannot be observed because the *PreservationObject* is corrupted or the required knowledge is incomplete. In this situation, *Property Values* are not measurable at the time because you lack information.

**Example:**

- An old Cyrillic font that is used in a document is not available on our machine configuration. An interesting discussion of this can be found in (Woods, Brown, 2011).

**Derivability:**

The *Property Values* in this category can, by definition, not be determined.

***Property* Categories that are Independent of Digital Objects but Important to Digital Preservation**

There are additional *Property* types that are independent of *PreservationObjects*, but they still affect *PreservationServices*.

- Representation Independent Properties

There are preservation *Properties* that are independent of the *File*, *Representation* or *Environment* (rendering stack).

There may, for example, be a *Constraint*

*"If a preservation action is chosen, it must be either a migration or a data refresh. Other preservation action types are not supported."*

This *Constraint* guides the preservation plan by specifying the *Property preservationActionType*, but does not refer to *Properties* which could be extracted from digital objects.

- User Experience Properties

Different users experience (see Figure 32) the same performance of a *PreservationObject* differently. E.g. somebody who participated in a competition will perceive images documenting the event different from somebody who was not involved or who does not understand the rules underlying the competition. These are *Properties* that describe the stakeholder's experience rather than the system's performance – those that relate to the psychological effect of object characteristics on a stakeholder.

This category is different from the 'Non-Predictable *ValueOrigins*' category discussed above, since it considers emotional impact rather than how the *Value* is obtained.

The analysis shows where one can push the boundaries of automation to compute *Properties*. It supports the argument that incomplete, approximate and heuristic *Values* need to be accommodated. It illustrates why there is a need for an expression language for *Properties* to define derived *Properties*. It also illustrates why there is a need for robust aggregate comparisons of digital object *Property Values*. Finally, it argues that there is a need to capture the semantics of similar *Properties*.

**Modelling Requirement 1.3.7:**

There can be multiple ways of obtaining the *Value* of a *Property* since there may be several representations (sources) which form the basis of measurement for the value, and several different measurement techniques (technique) and tools (or creation agents). As long as they do not produce conflicting results they all apply to the same *Property*. If they produce different results[22] then one should create related, but different *Properties* associated with them. If the *Value* for a given *ValueOrigin* has systematic differences that are related in a deterministic way to the *Value* of a different *ValueOrigin* then this difference should be recorded with their associated, related *Properties*.

If there are multiple ways of obtaining its *Value*, a *Property* can belong to several of the categories described in Modelling Requirement 1.3.6. For example, *imageWidth* can be extracted from a *File* (category 'Extractable, File-Based ValueOrigin'), calculated from other *Properties*, such as *resolution* and *pixelCount* (category 'Inferable ValueOrigin'), obtained from the rendering software (category 'Extractable, Complex ValueOrigin'), or measured by hand from a printed sheet (category 'Non-Predictable ValueOrigin'). *authorName* can be extracted from XML mark-up, HTML headers, MS Windows *File Properties*, etc. (category 'Extractable, File-Based ValueOrigin') or entered by hand (category 'Manually Assigned ValueOrigin'). *lineLength* can be extracted from a vector graphic (category 'Extractable, File-Based ValueOrigin') or calculated through heuristic algorithms based on a raster representation of the line (category 'Implicit Semantics ValueOrigin').

One important task of a *Property* ontology is to capture those *ValueOrigins* and their relationships.

[22] Modulo differences in their *Units*

**Modelling Requirement 1.3.8:**

*Property* ontologies have to deal with the semantics of related *Properties* so that they can be compared or derived from each other. This can be used to overcome the clashes between different *PreservationServices* that were observed in section 3.1.4.1. From the preceding analysis, it can be observed that *Properties* that are related to each other functionally (e.g. through a *ValueOrigin* definition in the 'Inferable ValueOrigins' category), can be related to each other through this definition within or across *PreservationServices*.

In all situations of clash, *Properties* that are derived through non-repeatable *ValueOrigins* (e.g. through a *ValueOrigin* definition in 'Non-Predictable' and 'Time-Varying ValueOrigins' categories), cannot reliably be compared to other *Properties* through simple equality metrics. They may be assessed with complex comparison metrics.

Properties that are non-determinable, e.g. in the 'Indeterminable ValueOrigins' category, cannot be compared to others.

**Modelling Requirement 1.3.9:**

*Characteristics* and *Constraints* need to specify on which *Property* they are based so that *Values* are not inadvertently compared for equivalence if their *ValueOrigins* produce non-equivalent *Values*.

**Modelling Requirement 1.3.10:**

*Values* for *Properties* can be obtained automatically or manually. Much research has gone into automatically extractable *Properties*. For large volumes of objects, manual declaration of *Property Values* by means of free format texts is unworkable. Unfortunately, it is evident that a large set of *Properties* that users require can be extracted automatically only with great difficulty or not reliably. There is a justified desire, where possible, to capture relationships such that most *Characteristics* can be automatically inferred from automatically extractable *Characteristics*. However, as the *imageWidth* and *authorName* examples above illustrate, whether or not a *Property* is obtained automatically is an orthogonal issue to their categories in Modelling Requirement 1.3.6.

### 3.1.4.3 Units

Definition of Unit

A *Unit* is a determinate quantity (as of length, time, heat, or value) adopted as a standard of measurement of which the magnitudes of other quantities of the same kind can be stated.

Modelling Requirements for Unit

**Modelling Requirement 1.3.11:**

Several *Unit*s and data constraints can apply to a *Property*. This is particularly important for preservation characterisation. *bitDepth*, for example, is described as one non-negative number in .png and as three non-negative numbers (one for each colour channel) in .tiff. It is important to be able to specify which data constraint is chosen and also, how this data constraint can be compared to others.

**Modelling Requirement 1.3.12:**

A *Property Value* can be represented with various *Units*. The *Value* for a *Property* of a given *Unit* can be converted deterministically to a different compatible *Unit*. *Unit* ontologies can be found in the Sciences (for example in QUDT (Hodgson, Keller, 2011) and search engines (for example in unit-ontology (OBO Foundry Initiative, nd)).

### 3.1.4.4 Characteristics

Definition of Characteristic

A *Characteristic* of an entity is the concrete *Value* which this entity has for an abstract *Property* in a defined context (a concrete *Property/Value* pair).

In the model it is the *Characteristic* of a *PreservationObject*, *Environment* or *PreservationAction*.

Modelling Requirements for Characteristic

**Modelling Requirement 1.3.13:**

In this model, each of the entities *PreservationObject*, *Environment*, and *PreservationAction*, may have *Characteristics*. This is a key aspect of this model.

*PreservationObjects* may have *Characteristics*.

**Examples:**

- *alignment*= “left” is a *Characteristic* of a *TextComponent*.
- *semanticInterpretation*=”body weight” is a *Characteristic* of a *Number-Component.*

*PreservationActions* may have *Characteristics*.

**Example:**

- *numberOfIntermediateCopiesProduced* = 2 is a *Characteristic* of a *PreservationAction*. It may, for example, be used to identify *PreservationActions* which violate copyright regulations that limit the number of intermediate copies created.

*Environments* may have *Characteristics*.

**Examples:**

- *memoryUsage* = “low” is a *Characteristic* of a *SoftwareToolEnvironment* that renders the *PreservationObject*.
- *numberOfIntermediateCopies* <=3 and *preservesColourDepth* = “yes” are *Characteristics* of a *PreservationTool* which is part of a *PreservationAction*’s *Environment*.[23]

It is essential to always be clear with which entity the *Characteristic* is associated, i.e. for which (in the general case, vector of) *PreservationObject*, *Environment* or *PreservationAction* this *Characteristic* holds.

**Modelling Requirement 1.3.14:**

*Characteristics* are used to express *Constraints* which then inform the choice of *PreservationAction*.

[23] They are class *Characteristics* which can be captured in a *PreservationServices* registry. If the service *Characteristic* reflects constant behaviour and the *ValueOrigin* can be trusted, it can be inherited to the *PreservationActions* that are executed by this service. In that case, the *Characteristic colourDepth* need not be measured and compared for individual *PreservationActions* since it is known to be preserved beforehand.

## 3.2 The full conceptual model

The full conceptual model extends the core model with concepts from the digital preservation domain: *PreservationRisk*, *PreservationAction*, *Constraint* and *Policy*, as illustrated in Figure 33. It shows the entities and relationships which are explained in detail in the following sections.

An essential aspect of this model is that it takes into account the goals and limitations of the stakeholder, features of its user community, and the environment in which its users access digital content. Thus, the scope of digital preservation extends beyond merely considering file formats and preserving *Characteristics* of individual digital objects.

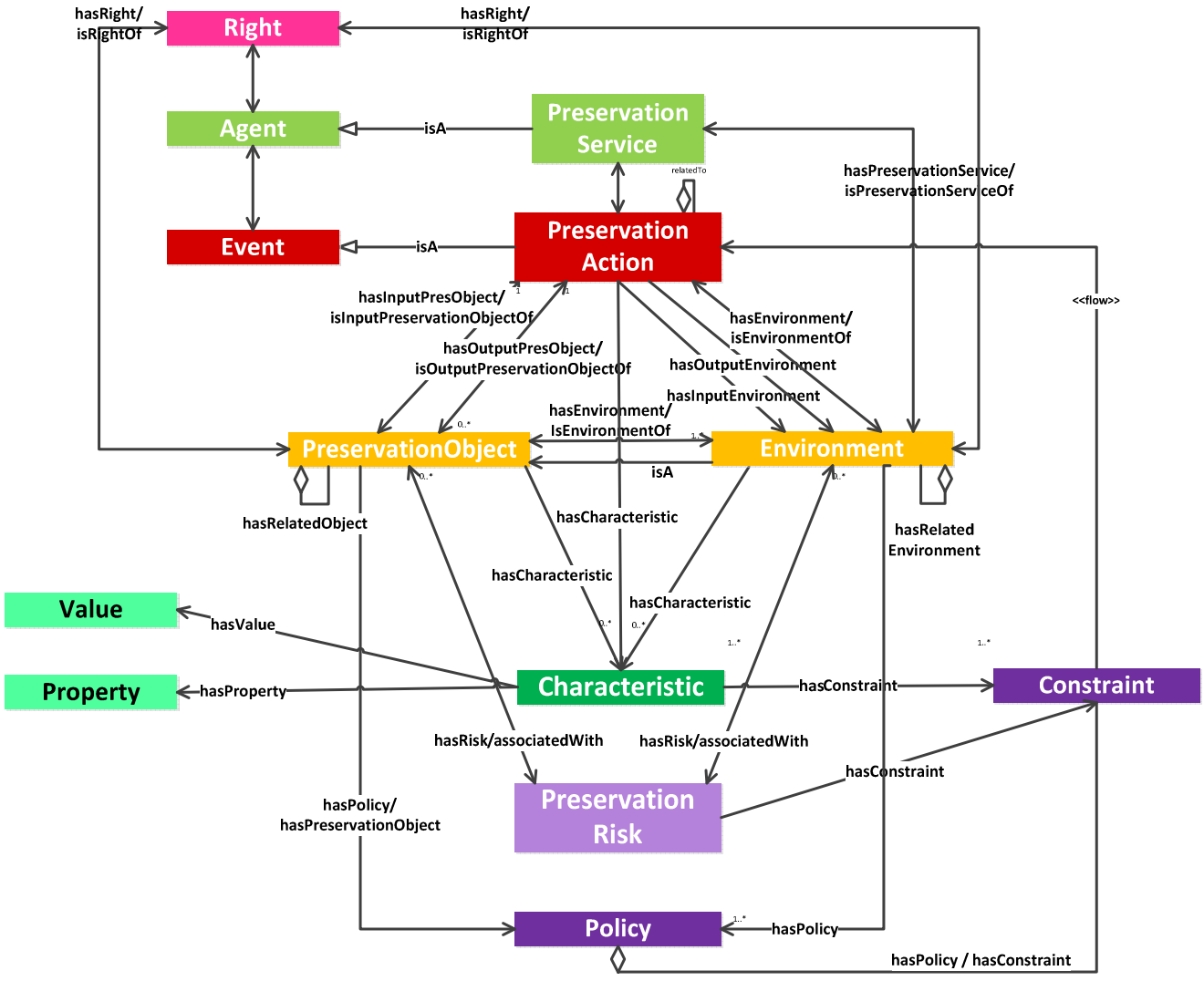



**Figure 33: Full conceptual model**

Degradation of *PreservationObjects* is caused by two things:

- *PreservationRisks*
- Executing imperfect, lossy *PreservationActions*

Acceptable levels of degradation are defined in an institution's *Constraints*, which specify permissible or desirable *Characteristics* of *PreservationObjects* and *Environments*. They make the institution's values explicit, influence the preservation process, and are captured in *Policy* documents.

Changes to a *PreservationObject* or *Environment*, such as obsolescence of hardware or software components, decay of data carriers, or changes to the legal framework may introduce *PreservationRisks*. An individual institution's *PreservationRisks* are specified in *RiskSpecifyingConstraints*. Whenever *Characteristics* of a *PreservationObject* or its *Environments* violate the *RiskSpecifyingConstraints*, then the *PreservationObject* is considered at risk. Once a *RiskSpecifyingConstraint* is violated, a preservation monitoring process should notice this and trigger the risk mitigation / preservation planning process. It, in turn, determines the best *PreservationAction* to mitigate this risk. *PreservationObjectSelectingConstraints* are a sub-class of *RiskSpecifying-Constraints* which specify which subset of *PreservationObjects* is at risk.

A composite *PreservationAction* may consist of elementary *PreservationActions.*

When a *TransformationPreservationAction* is applied to a *PreservationObject* and its *Environment*, it produces a new *PreservationObject* and/or a new *Environment* in which the *PreservationRisk* has been mitigated. Every *TransformationPreservationAction*, therefore, has not only an input *PreservationObject* and (at least one) input *Environment*, but also an output *PreservationObject* and output *Environment* as seen in Figure 34. For example, if a Microsoft Word *File* is migrated to a .pdf *File*, this results in a new *PreservationObject*, which has different *Characteristics*, but also a new *Environment* in which it can be used – in this case the platform needs at least to contain a .pdf viewer. This approach works for migration, emulation, hardware and other solutions.

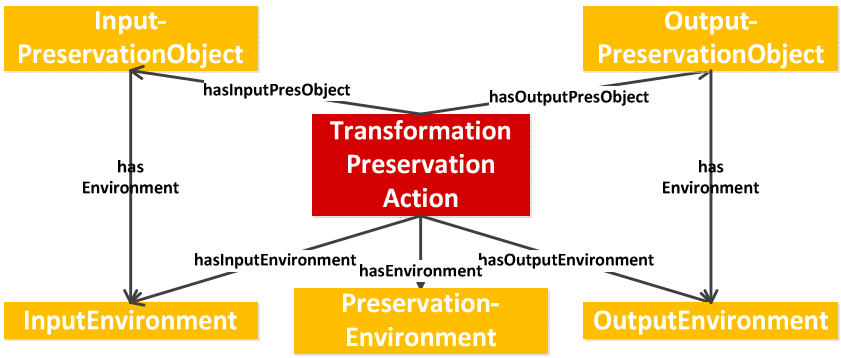



**Figure 34: *TransformationPreservationAction*'s relationships**

For any given *PreservationObject* and its *Environment*, there may be multiple possible *PreservationActions* to mitigate a *PreservationRisk*. Which of these *Preservation-Actions* is the most suitable for the *PreservationObject* can be derived from the information in the *Constraints*. In order to determine whether an abstract *Constraint* is applicable and satisfied, one needs to evaluate the concrete *Values* of the *Characteristics* of *PreservationObjects* and their *Environments* or the concrete *Values* of a candidate *PreservationAction* at a given time.

Some *Constraints* can be expressed in a machine-interpretable way. They refer solely to concepts and vocabulary contained in the model. They may include a conditional context, pre- and post-conditions, and sometimes complex expressions. In addition, it is useful to specify the relative importance and acceptable tolerances for *Constraints*. Importance factors specify the importance of a *Constraint* for an institution. A tolerance threshold specifies the degree to which deviation from the *Constraint* can be accepted.

*Events*, *Agents* and *Rights* are entities in the model and may be taken from PREMIS (2012).

### 3.2.1 Digital preservation risks

Definition of PreservationRisk

A *PreservationRisk* arises when a *Characteristic* of a *PreservationObject* or of an *Environment* of a *PreservationObject* conflicts with the stakeholder's *RiskSpecifying-Constraints*.[24]

Modelling Requirements for PreservationRisk

**Modelling Requirement 2.1.1:**

The goal of digital preservation is to mitigate *PreservationRisks* to *PreservationObjects* (or to take advantage of opportunities for improvement) through *PreservationAction*s.

**Modelling Requirement 2.1.2:**

Specific *PreservationRisks* are associated with a vector of *PreservationObjects* or *Environments* of a *PreservationObject*.

**Examples of *PreservationRisk* include:**

- Data carriers deteriorate and cannot be read.
- The data object becomes corrupted on the carrier and the original *Bytestream* cannot be retrieved.
- Essential hardware components are no longer supported or available.
- Software components are proprietary and this dependence is unacceptable to the stakeholder.
- The community requires new patterns of access, such as access on a mobile phone, rather than a workstation.
- File formats become obsolete.
- The legislative framework changes and the data or access to it has to be adapted to the new regulations.

[24] This thesis does not distinguish between risks (things that may happen) and issues (things that have happened). Nor does it distinguish between threats and opportunities. In consequence *Preservation-Actions* include proactive and reactive actions.

**Examples of *PreservationOpportunities* include:**

- Adding features, such as interactivity, provides new usage opportunities.
- Maintaining data becomes cheaper by moving to alternative formats.
- Consolidating support structures (e.g. software or hardware *Environments*) streamlines the maintenance of the *Collection*.

In the remainder, the term *PreservationRisks* implicitly includes *Preservation-Opportunities*.

**Modelling Requirement 2.1.3:**

*PreservationRisks* are not inherent, but are relative to considerations such as the stakeholder's requirements, as captured in *Constraints*, and the *Characteristics* of *PreservationObjects* and *Environments*.

**Examples:**

- Depending on the stakeholder's requirements: One stakeholder might find using proprietary software acceptable, another might not, and, therefore, does or does not consider it a *PreservationRisk.*
- Depending on the digital object's *Characteristics*: The digital object uses, or does not use macros and, therefore, is or is not subject to a *PreservationRisk*.

Each stakeholder must, therefore, specify in *RiskSpecifyingConstraints* which state of the *PreservationObject* or the *PreservationObject's Environment* represents a *PreservationRisk*.

**Modelling Requirement 2.1.4:**

Risks apply to technological *Environments*. But they also apply to community *Environments*. If, for example, consumers request changed services (i.e. they consider existing services obsolete) then this may prompt the need for executing a *PreservationAction* which brings the services up to date.

Vocabulary for PreservationRisk sub-classes

The *PreservationRisk* class is extensible. There are many possible sub-classes for *PreservationRisks* that can prove to be useful for different digital preservation contexts.

**Examples:**

Drambora (McHugh, Innocenti, Ross, 2008) suggests a breakdown structure by

- preservation functions (commitment to digital object maintenance, organisational fitness; legal & regulatory legitimacy; effective & efficient policies; acquisition & ingest criteria; integrity, authenticity & usability; provenance; dissemination, preservation planning & action; adequate technical infrastructure) or by
- constraint type (technological, physical, organisational, socio-cultural, legal, economic, financial, political, contractual, environmental).

Barateiro et al. (2010) break *PreservationRisks* down by

- vulnerabilities (Process (software faults, software obsolescence); data (media faults, media obsolescence), infrastructure (hardware faults, communication faults, network services failures)) and
- threat sources (disasters (natural disasters, human operational errors), attacks (internal attacks, external attacks), management (economic failures, organisational failures), legislation (legislative changes, legal requirements)).

In the ISO 16363 Standard for Trusted Digital Repositories (CCSDS, 2011) risk categories are divided into

- organisational infrastructure (governance & organisational viability, organisational structure & staffing, procedural accountability & preservation policy framework, financial sustainability, contracts, licenses, & liabilities),
- digital object management (ingest: acquisition of content, ingest: creation of the AIP[25], preservation planning, AIP preservation, information management, access management),
- infrastructure and security risk management (technical infrastructure risk management, security risk management).

The following *PreservationRisk* categories specific to the *PreservationObject* and its *Environment* were identified during policy document analysis (as illustrated in Figure 35):

[2525] Archival Information Package

- *NewVersionRisk*: A new version of the *PreservationObject* or *Environment* is available. This creates a risk of future obsolescence, or a risk of having to support too many versions.
- *LackingSupportRisk*: The *PreservationObject* or *Environment* is no longer sufficiently supported. This creates a risk that support will cease altogether, rendering the *PreservationObject* or *Environment* inaccessible.
- *DeteriorationOrLossRisk*: The *PreservationObject* or *Environment* is deteriorating or has been lost. Reconstruction or replacement become necessary.
- *ProprietaryRisk*: The *PreservationObject* or *Environment* is proprietary. There is a risk that it cannot be replaced since the specifications for it are unknown.
- *UnmanagedGrowthRisk*: The stakeholder's *PreservationObjects* or *Environments* are becoming too diverse to manage. A "normalisation" *PreservationAction* is needed to simplify or unify them.

These risk categories can be used to create sub-classes of *RiskSpecifyingConstraints*.

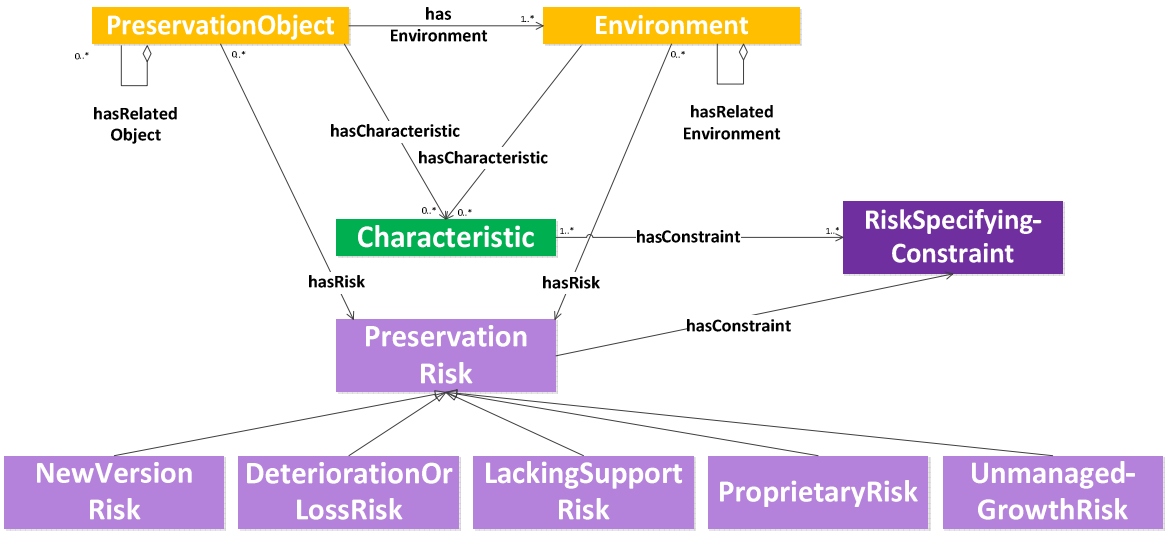



**Figure 35: Example vocabulary for *PreservationRisk* sub-classes**

### 3.2.2 Digital preservation services and actions

Custodians of digital content take *PreservationActions* to mitigate the *Preservation-Risks* that they identify. A *PreservationAction* (*Event*) takes place when a *Preservation-Service* (*Agent*) is invoked. A *PreservationService* is an *Agent* that provides a core service supporting the goal of digital preservation. A *PreservationAction* is applied to an existing, input *PreservationObject* and *Environment*. If the *PreservationAction* is a transformation (i.e. it results in a state change) it results in either a new output *PreservationObject* and/or a new *Environment*. Together they mitigate the *PreservationRisks* that the *PreservationAction* addresses. For example, a Microsoft Word *File* is migrated to a .pdf *File* in order to lock in the desired look-and-feel of the document. The output *Environment* must support a .pdf viewer. *Characteristics* of the output *PreservationObject* and the output *Environment* are validated against *SignificanceCharacteristics* (a type of *Constraint*) in order to quantify the degree of compliance. This approach to describing *TransformationPreservationActions* works for migration, emulation, hardware replacement, and other solutions.

Definition of PreservationService

A *PreservationService* is an Agent that provides a core service supporting the goal of digital preservation.

Examples are preservation risk monitoring; determining *Characteristics* of *Preservation-Objects* and *Environments*; comparison of *Characteristics* to determine authenticity; and planning, execution and evaluation of candidate *PreservationActions*. *PreservationServices* are realised manually or through software tools and are provided through software, hardware, human and other *Environments*.

A *PreservationService* is an *Agent* that executes an *Environment* (*Tool*).

Definition of PreservationAction

A *PreservationAction* is an Event resulting from the execution of a *PreservationService*. The execution of a *PreservationService* that mitigates a *PreservationRisk* to the continued viability, renderability, understandability, and authenticity of a *PreservationObject* across time and changing *Environments*. It ensures the satisfaction of their *Constraints*. A *TransformationPreservationAction* may transform the *PreservationObject* itself, the *Environment* required to support access to the *PreservationObject*, or a combination thereof.

A *PreservationAction* is an *Event* resulting from the execution of a *Preservation-Service*.

Modelling Requirements for PreservationAction

**Modelling Requirement 2.2.1:**

A *TransformationPreservationAction* produces a changed version of the *PreservationObject* and/or its *Environment*. The model, therefore, contains an input and output *PreservationObject* and input and output *Environments* for a *Transformation-PreservationAction* (Figure 34).

**Examples:**

- In the case where a corrupted *File* is recovered from a back-up, there is an input and output *File* while the *Environment* may stay the same.
- In the case of migration, there is an input and output *Representation*. The input and output *Representations* may need different *Environments*.
- In the case of data carrier refresh, the input and output *Files* are the same, but the *Environment* is new.

**Modelling Requirement 2.2.2:**

A *TransformationPreservationAction* produces a new *PreservationObject*, if the intellectual content of the *PreservationObject*, the semantic and syntactic interpretation of the content which are necessary to interpret the content, the format in which the content is encoded, or the physical realisation of the content change.

**Example:**

In the case of file reconstruction there is an input and output *File* since the realisation of the *File* changes. If the *File* is part of a *Representation*, then there will also be a new output *Representation* object, or possibly even a new *IntellectualEntity* if *Characteristics* change sufficiently.

**Modelling Requirement 2.2.3:**

In general a *TransformationPreservationAction* may result in the replacement or repair or reconstruction of a combination of *Environments*.

**Example:**

Emulation can be seen as a combination of hardware, software and file format replacement, since it provides a new hardware and/or software *Environment* for the digital object, but it might also be necessary to extract data from the original digital object to feed into the emulation.

**Modelling Requirement 2.2.4:**

Input and output *Representations* of *TransformationPreservationActions* may consist of several *Files*.

**Examples:**

- Several input files: When migrating an .xml *Representation* to a .pdf *Representation*, the input *Representation* consists of the .xml *File* and its images. Migrating an Oracle database to an Access database, consumes .dbf, .ctl *Files*, etc. and produces one .mdb *File*.
- Several output files: When migrating a Microsoft Word *Representation* to an HTML *Representation*, the output *Representation* consists of the .html *File* with an accompanying .css *File*. Migrating a .zip *File* to its expanded version leads to multiple formats.

**Modelling Requirement 2.2.5:**

Every *PreservationAction* is associated with the *Environment* required for its own execution. The *Hardware* on which the action is executed and the *PreservationService* that is invoked (e.g. a certain configuration of a migration tool), for example, are parts of this *Environment*.

**Modelling Requirement 2.2.6:**

*PreservationActions* may have *Characteristics* of their own. They may be used to identify *PreservationActions* that violate *ActionDefiningConstraints* that define which kinds of *PreservationActions* are desirable, or they are used to express *PreservationGuidingConstraints* which are conditional on *Characteristics* of *PreservationActions*. (see section 3.2.3.2.)

**Examples:**

- ***numberOfIntermediateCopiesProduced*** = 2 is a ***Characteristic*** of a *TransformationPreservationAction*. It may be used to identify *TransformationPreservationAction* that violate ***Constraints*** that specify copyright regulations or license agreements that limit the number of intermediate copies created.
- ***acceptedInputFormat*** and ***outputFormats*** of the associated ***PreservationService*** are ***PreservationAction Characteristics.***
- ***preservationActionCost*** is a ***Characteristic*** of a ***PreservationAction.***

Vocabulary for PreservationAction sub-classes

The *PreservationAction* class is extensible. There are many possible sub-classes for *PreservationActions* that can prove to be useful for different digital preservation contexts. Such *PreservationAction* sub-classes may suitably be described in a registry.

The following *TransformationPreservationAction* categories were identified during policy document analysis: A *TransformationPreservationAction* may result in the *Replacement*, *Repair* or *Reconstruction* of any of the *PreservationObjects* or *Environments* that are at risk. This is illustrated in Figure 36.

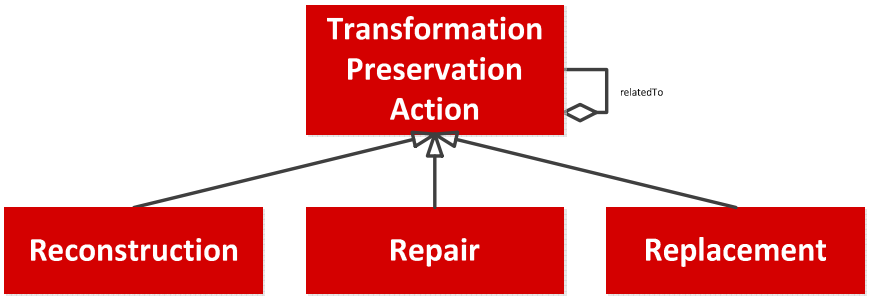



**Figure 36: Vocabulary for *TransformationPreservationAction* sub-classes**

During preservation planning, every combination of *PreservationObject*, and *Environment* can be matched to appropriate *PreservationActions* to mitigate a given *PreservationRisk*.

**Examples:**

- The risk of data carrier failure can be mitigated by a carrier refresh.

- The risk of file format obsolescence can be mitigated by migrating objects to an alternative format.

The diagram (Figure 37) and table (Table 5) illustrate some *TransformationPreservation-Action* sub-classes depending on

- sub-class of the affected *PreservationObject* and/or *Environment*
- *PreservationRisk* sub-class

**Table 5: Examples of *TransformationPreservationAction* sub-classes**

| Example Risks | Preservation-Object sub-class | Environment sub-class | Preserva-tionRisk sub-class | Preservation Action sub-class |
|---|---|---|---|---|
| Data carriers deteriorate and cannot be read | | Storage Medium | Deterio-ration | Replacement - carrier refresh |
| The digital object becomes corrupted on the carrier and the original Bitstream cannot be retrieved. | Bitstream | | Deterio-ration | Reconstruc-tion |
| Essential hardware components are no longer supported or available | | Hardware | Lacking support | Replacement |
| Software components are proprietary and the dependence is unacceptable to the institution. | | Software | Proprietary | Replacement |
| The community requires new patterns of access, such as access on a mobile phone, rather than a workstation | | Hardware and Software | Obsolete | Replacement |
| File formats become obsolete. | File | | Obsolete | Replacement - migrating objects to an alternative format |
| The legislative framework changes and the data or access to it has to be adapted to the new regulations | | Legislation | New Version | Replacement |

**Examples:**

Figure 37 shows some examples of *TransformationPreservationAction* sub-classes depending on the sub-classes of the *PreservationRisk* and the affected *PreservationObject* or *Environment*.

Most of them are self-explanatory. Some deserve some special comments:

- Modification of *Content* might represent a *PreservationAction* such as the reconstruction of a deteriorated *File*, or a *File* that is modified in order to satisfy new legal *Constraints*.
- One possible *PreservationAction* is not to do anything ("wait and see").
- Migration does not always imply that a different file format is chosen. One might, for example replace an .xml *File* with another .xml *File*. In that case the input and output file formats happen to be the same. The output *PreservationObject* might nonetheless have different *Characteristics* to the input *PreservationObject* because of the different information captured within the xml tags.
- The needs of the target community might be a deciding factor for the choice of *PreservationActions*, and, conversely, the choice of *PreservationActions* will shape and change the community, just as it changes the other *Environment* sub-classes.
- Community consists of producers and consumers. Both types are either technical (e.g. repository or IT staff, publishing staff) or content oriented (authors or readers) and will consider the digital object obsolete under different circumstances and according to their needs.
- Shifting the target community might be a somewhat unintuitive *PreservationAction*, which is parallel to all other forms of *Environment* replacement. An example might be turning a research data collection into a history-of-science repository, as the material contained in the collection ceases to live up to contemporary standards of scientific use.

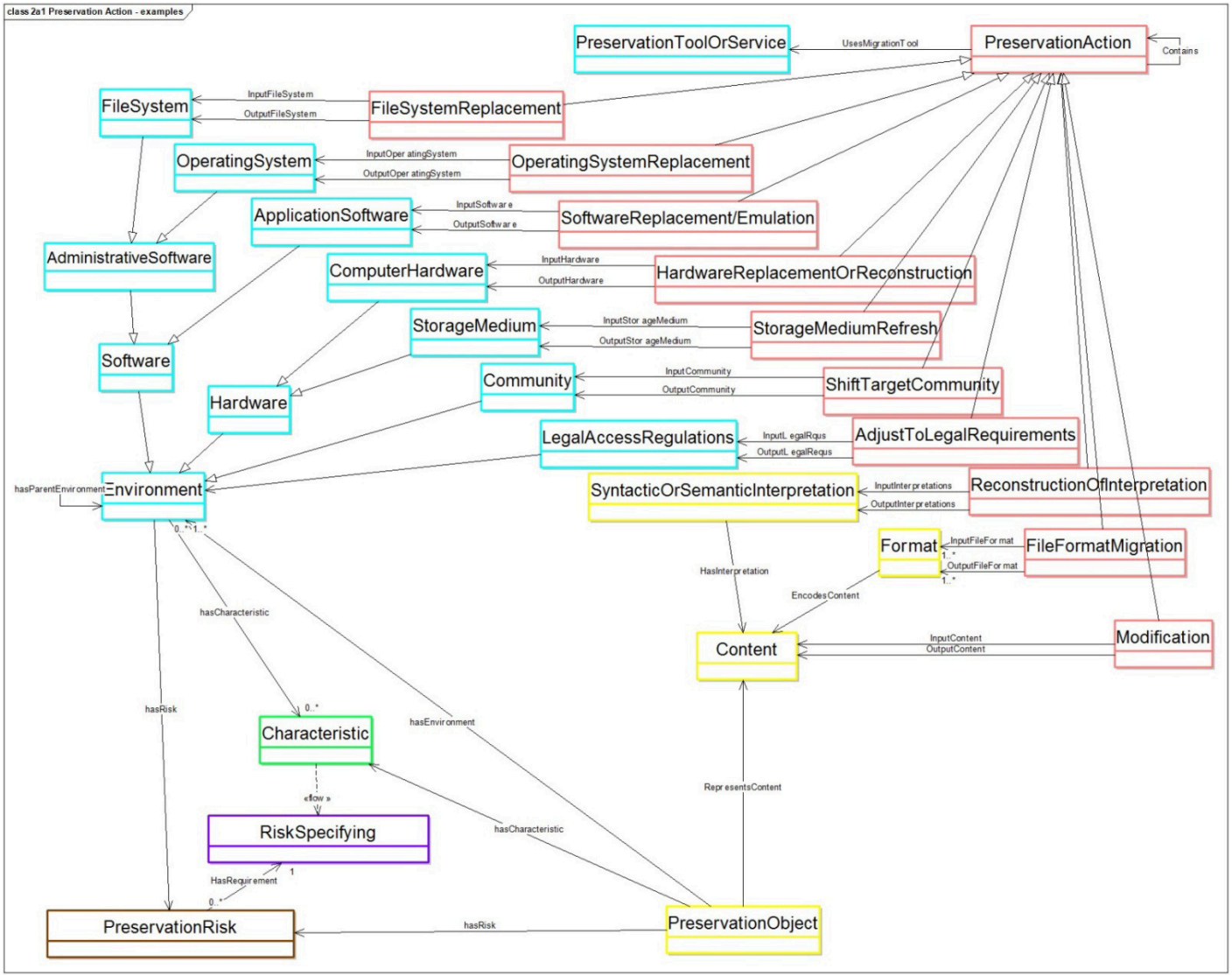



**Figure 37: Example *TransformationPreservationAction* sub-class depending on the sub-classes of *PreservationObjects* or *Environments* and of *Risk***

### *3.2.3 Constraints*

Definition of Constraint

A limitation or restriction on the space of allowable *PreservationActions*.

Modelling Requirements for Constraint

As opposed to models, such as PREMIS (2012) or OAIS (CCSDS, 2012) the model in hand recommends that *Constraints* or business rules should in the general case be represented as explicit top-level entities in a data model. Figure 33 introduces this separate concept.

**Modelling Requirement 2.3.1:**

*Constraints* make the stakeholder's values explicit and influence the digital preservation process.

*Constraints* are measurable subsets of goals. They express a target level of results expressed in units against which achievement is to be measured. *Constraints* provide the day-to-day support for achieving goals. (adopted from StratML, Objectives (StratML, nd))

**Modelling Requirement 2.3.2:**

*Constraints*

- define which input *Characteristics* of the *PreservationObject* and its *Environment* need to be met to consider a *PreservationAction*.
- define acceptable output *Characteristics* of the *PreservationObject* and its *Environment* for *TransformationPreservationActions*.
    - They may be dependent on input *Characteristics* by comparing the differences between the input and output *Characteristics* and measuring to what degree this difference satisfies the required *Characteristics*[26].

**Examples**:

- The loss of resolution may not exceed 20% of the original resolution.

[26] Because of chains of migration over time the input *PreservationObject* and its *Environment* might be a derivative of the original submitted to the stakeholder. In order to not accumulate errors in subsequent *PreservationActions* it is best to express comparative losses with respect to the original *PreservationObject*.

        - The size of the *TransformationPreservationAction's* output *PreservationObject* should be in a specified relationship to that of the input *PreservationObject*.
    - They may be expressed in absolute terms, independent of input *Characteristics* by measuring to what degree the output *Characteristic* satisfies the required *Characteristic*.

    **Examples**:

        - The size of the *PreservationAction*'s output *PreservationObject* should not exceed a maximal size set by the stakeholder.
        - Output file formats need to be platform independent.

- define acceptable *Characteristics* of the *PreservationAction* itself.

    **Example**:

        - *PreservationAction* tools must be open-source.

- describe the preservation process itself independent of the *Characteristics* of the *PreservationObject* as well as of those of the *PreservationAction*.

    **Example**:

        - A preservation planning process should be executed for every data object at least every 5 years, independent of the *PreservationRisks* that are established for this data object.

**Modelling Requirement 2.3.3:**

*Constraints* are captured in *Policies*. This model uses the term *Policy* to include a variety of documentation, in a broad sense. They may be policy, strategy or business documents, applicable legislation, guidelines, rules, or even a choice of temporary runtime parameters. They may be oral representations as well as written representations in databases, source code, websites, etc.

**Modelling Requirement 2.3.4:**

*Constraints* can be expressed through a formal constraint language, such as the Object Constraint Language (OCL) (Warner, Kleppe, 2003) or other informal or formal languages. They

can be expressed through one or more *Property/Value* constraint specifications on *PreservationObjects*, *Environments* or *PreservationActions* and any of their sub-classes.

**Modelling Requirement 2.3.4a**

In many cases, a stakeholder would like to make *Constraints* dependent on additional conditions - that is to say that a context needs to be specified. The conditions involve *Characteristics* of *PreservationObjects*, *Environments* or *PreservationActions*.

**Examples**:

- If *componentType* = "text" then *fontSize* must be preserved.
- If *environmentType* = "archival preservation" then *imageResolution* must be preserved .
- If *preservationActionType* = "bitPreservation" then *fileSize* must be preserved.

As a result, the language used to define *Constraints* must be expressive enough to include conditionals**.**

**Modelling Requirement 2.3.4b**

*Constraints* often need to include specifications such as invariants, pre-conditions and post-conditions.

**Modelling Requirement 2.3.4c**

A stakeholder may only instantiate consistent, non-contradictory sets of *Constraints*.

**Modelling Requirement 2.3.5:**

Not all *Constraints* are equally important and not all have to be precisely satisfied. To accommodate this, it is useful for a stakeholder to add an *ImportanceFactor*, as a measure of relative importance of the *Constraint* for the stakeholders. If each of two conflicting goals are considered significant one needs to prioritise one as more significant than the other. This prioritisation is essential for both decision making and planning.

**Example**:

- Preserving the number of lines on a page is less important than preserving the number of pages.

Additionally, *Constraint*s may tolerate some deviation or error. The model should have a *ToleranceFactor*, as a measure of the tolerable degree of deviation from the required *Value*, with each *Constraint*.

**Example**:

- An office document migration that produced a result with different hyphenation or pagination might be acceptable in many situations.

During *Constraints* evaluation of a *PreservationAction* the importance and tolerance factors can be combined into a weighted measure.

**Modelling Requirement 2.3.6:**

While *Characteristics* capture *Values* at a given moment in time, *Constraints* are parameterised and capture *Characteristics* across time – before and after a *PreservationAction*.

**Modelling Requirement 2.3.7:**

*Constraint* evaluators (e.g. the XCDL comparator (Thaller et al., 2008; Thaller, 2009)) determine the degree to which *Characteristics* of the *PreservationObject* and *Environment* before and after the execution of a candidate *PreservationAction* comply with the stakeholder's *Constraints*.

During preservation planning one determines to what degree candidate *PreservationActions* satisfy the combined set of *Constraints* and concludes from this which of the candidate *PreservationActions* is the most suitable. This process amounts to a cost/benefit analysis.

**Modelling Requirement 2.3.8:**

The output *Characteristic* of a *TransformationPreservationAction* is not necessarily inferior to the input *Characteristic*, i.e. preservation is not always lossy. In many cases, stakeholders wish to include the possibility of capturing improvements to a *PreservationObject*.

**Example**:

- A common *PreservationAction* is "normalisation" of digital *PreservationObjects* upon ingest. This may be done to reduce the variety of formats held, but may also be done to improve *Characteristics* in the original. For example, one might migrate *Files* which are in formats that are susceptible to degradation to *Files* in a more resilient format, or move static tables to spread sheets which enable pivot tables. In this case the *Characteristics fileFormatResilience* = "high" or *enablesPivot-Tables* = "yes" are significant *Characteristics (SignificanceConstraints)* which were not found in the original.
- Another *PreservationAction* which improves upon the original is the manual restoration of a *File* by a curator to the state it was presumed to have had before a corruption.
- Another common example can be found in CAD drawings or data sets. As technology improves, consumers desire to perform new functions on old data in ways that were previously not possible.

Example of Constraint

The following example in Figure 38 illustrates how a *Constraint* may be expressed solely in terms of model elements and vocabulary. They are taken from the model's conceptual detail definition in appendix 7.1.

The *Constraint* "Textual data must be migrated to RTF 1.8" is being mapped in the following way:

- The context of the *Constraint* describes the Class to which the precondition, post-condition, or invariant applies. In this example it describes restrictions on eligible *PreservationActions*.
- The precondition describes under which circumstances the *Constraint* applies. This is expressed solely in terms of the *hasInputPreservationObject* relationship between *PreservationAction* and *PreservationObject*, and in terms of the *hasCharacteristic* element of *PreservationObject*.
- The post-condition, finally, describes which conditions need to be true after a *PreservationAction* is executed under the given circumstances. Again this is expressed using relationships and entities introduced in the above data model.

- **Context**:
- 
- *PreservationAction*: ***a***
- ***class-of (a): "replacement preservation action"***
- ***has****Input*PreservationObject***: i***
- ***has****Output*PreservationObject***: o***

- **Precondition**:
- 
- *PreservationObject*
- ***preservationObjectIdentifier: i***
- ***class-of (i): "File"***
- ***hasCharacteristic: x***
- 
- *Characteristic*
- ***characteristicIdentifier: x***
- ***associatedWith: (i)***
- ***has*Property***: p9067***
- ***hasValue: "text"***
- 
- *Property*
- ***propertyIdentifier: p9067***
- ***propertyName: "formatType"***
- ***appliesTo: (*Bytestream*)***
- ***range***
- ***hasDataConstraint: formatType vocabulary***
- ***hasValueOrigin:***
- ***hasValueOriginID: vo12756***
  (e.g. this might specify the software that characterises the formatType)

- **Postcondition**:
- 
- *PreservationObject*
- ***preservationObjectIdentifier: o***
- ***class-of (o): "File"***
- ***HasCharacteristic: y***
- 
- *Characteristic*
- ***characteristicIdentifier: y***
- ***hasObject: o***
- ***has****Property****: p782***
- ***hasValue: "fmt/53"***
  (this is the unique identifier (PUID) for RTF 1.8 in the PRONOM registry)
- 
- *Property*
- ***propertyIdentifier: p782***
- ***propertyName: "formatDesignation"***
- ***range***
- ***hasDataConstraint: PUID***
- ***hasValueOrigin:***
- ***hasValueOriginID: vo908***
  (e.g. this might specify the PUID look-up in the PRONOM registry)

**Figure 38: Example *Constraint***

Vocabulary for Constraint sub-classes

Different *Constraint* categories play different roles in the digital preservation process. During the literature and document analysis described in section 1.3.1, *Constraints* were extracted from the interview protocols and policy documents and were categorised into the sub-classes depicted in Figure 39. The knowledgebase of *Constraints* captured is published in (Dappert, Ballaux, Mayr, van Bussel, 2008).

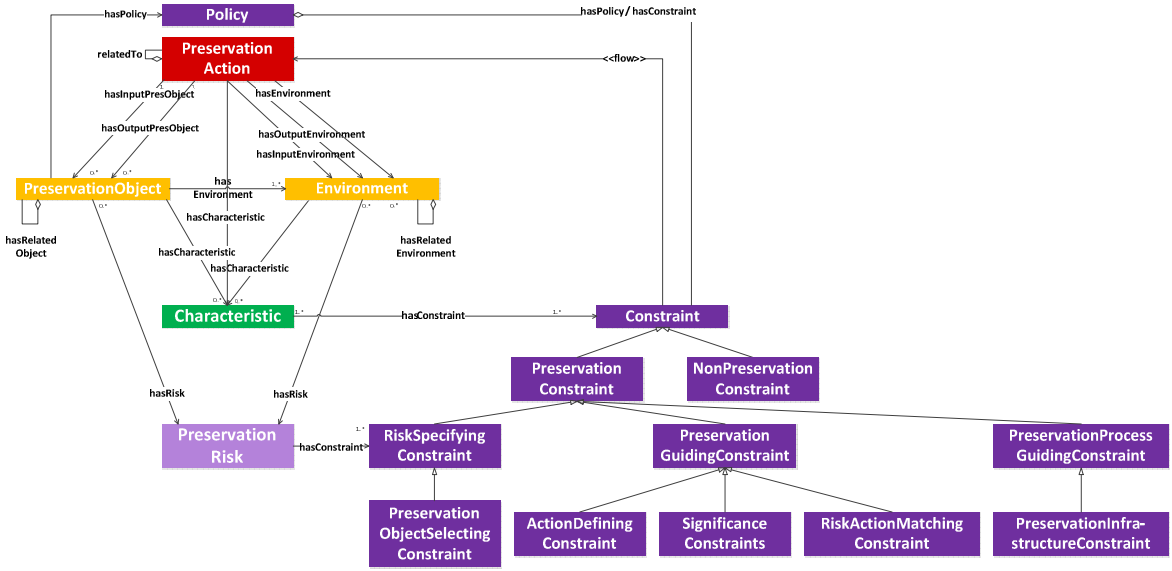



**Figure 39: *Constraint* sub-classes**

### *3.2.3.1 Constraints that specify risks*

#### *3.2.3.1.1 RiskSpecifyingConstraints*

*RiskSpecifyingConstraints* state explicitly what the perceived risks for *Preservation-Objects* and *Environments* are. Whenever *Characteristics* of a *PreservationObject* or its *Environment* violate a *RiskSpecifyingConstraint* then the *Preservation-Object* is considered at risk.

Once a *RiskSpecifyingConstraint* is violated, a preservation monitoring process should trigger the preservation planning process. It, in turn, determines the optimal *PreservationAction* which should mitigate this *PreservationRisk*.

**Examples**:

- The *licenseStatus* of the *RenderingSoftware* of the *PreservationObject* is "lapsed". This implies that the *PreservationObject* is considered at risk.

- Random sampling of a set of *PreservationObjects* shows more than 0.5% corruption. This implies that these *PreservationObjects* are to be considered at risk.

#### *3.2.3.1.2 PreservationObjectSelectingConstraints*

*PreservationObjectSelectingConstraints* are a special class of *RiskSpecifying-Constraints* which specifies sets of *PreservationObjects* that have specific *Characteristics* that influence the risk impact and, therefore, the risk mitigation.

**Examples**:

- The total number of *PreservationObjects* of a defined value in a set of *PreservationObjects* exceeds a threshold number. This implies that the collection is large enough to be considered of substantial value and large enough to justify the expense of a certain *PreservationAction*.
- *PreservationObjects* in a set do not have printed backups. This implies that these *PreservationObjects* are at increased risk and should be prioritised for *PreservationActions*.

**Modelling Requirement 2.3.9:**

The *ImportanceFactor* and *ToleranceFactor*, mentioned in Modelling Requirement 2.3.5 above, play for *RiskSpecifyingConstraints*, the role that impact metrics traditionally play in risk management activities (e.g. as used in Drambora (McHugh, Innocenti, Ross, 2008)) to help prioritise among several risk *Constraints*.

**Modelling Requirement 2.3.10:**

In order to structure the *RiskSpecifyingConstraints* set further, one can create sub-categories along the *PreservationRisk* sub-classes (see Figure 35) *NewVersion*, *NotSupportedOrObsoleteSupport*, *DeteriorationOrLoss*, *Proprietary*, and *UnmangedGrowth* if this distinction supports the preservation process.

### *3.2.3.2 Constraints that guide preservation actions*

#### *3.2.3.2.1 PreservationGuidingConstraints*

*PreservationGuidingConstraints* specify which kinds of *PreservationActions* are desirable with respect to a *PreservationObject* and its *Environments* by explicitly stating

the stakeholder's values. The degree to which the *PreservationAction* satisfies those *Constraints* determines its cost/benefit for the stakeholder.

**Example**:

- If the *PreservationObject* has the *Characteristic contentType* = "email", then the *TransformationPreservationAction* has to produce an output *PreservationObject* with *Characteristic fileFormat* = "XML".

#### *3.2.3.2.2 SignificanceConstraints*

*SignificanceConstraints* are a special class of *PreservationGuidingConstraints*. They define which *Characteristics* must be met by output *PreservationObjects* and *Environments*. Our definition of *SignificanceConstraint* is close to the one expressed by Andrew Wilson (National Archives of Australia) for "significant properties": "the *Characteristics* of digital objects that must be preserved over time in order to ensure the continued accessibility, usability, and meaning of the objects, and their capacity to be accepted as evidence of what they purport to record.". It is important to note that DePICT treats them as *Constraints* rather than as *Properties*. It considers *SignificanceConstraints* for any *PreservationObject* or *Environment* sub-class, not just for *PreservationObjects*. Because of the importance of *SignificanceConstraints*, section 3.2.3.5 analyses them in more detail.

#### *3.2.3.2.3 ActionDefiningConstraints*

*ActionDefiningConstraints* are a special class of *PreservationGuidingConstraints*. They define which kinds of *PreservationActions* are desirable independent of the *Characteristics* of the *PreservationObject*, but dependent only on the *Characteristics* of the *PreservationAction* itself.

**Example**:

- *PreservationAction* tools must satisfy the institution's software quality standards.

#### *3.2.3.2.4 RiskActionMatchingConstraints*

*RiskActionMatchingConstraints* are a special class of *PreservationGuidingConstraint*. They specify that a candidate *PreservationAction* has to be an appropriate match to a given *PreservationRisk* as was illustrated in Figure 37. They are rarely stated explicitly in *Policy* documents since this is assumed to be common sense.

#### 3.2.3.3 Constraints that guide the preservation process

##### 3.2.3.3.1 PreservationProcessGuidingConstraints

*PreservationProcessGuidingConstraints* describe the preservation process itself independent of the *Characteristics* of the *PreservationObject*, its *Environments*, as well as of those of the *PreservationAction*. They may prompt the preservation planning process but do not influence it.

**Example**:

- A preservation planning process should be executed for every data object at least every 5 years, independent of the *PreservationRisks* that are established for this data object.

##### 3.2.3.3.2 PreservationInfrastructureConstraints

*PreservationInfrastructureConstraints* are a special class of *PreservationProcess-GuidingConstraints* which specifies what *Characteristics* are required of the infrastructure with respect to security, networking, connectivity, storage, etc.

**Example:**

- Mirror versions of on-site systems must be provided.

#### 3.2.3.4 Constraints that impact preservation

##### 3.2.3.4.1 NonPreservationConstraints

*NonPreservationConstraints* are a special class of *Constraints*. They specify processes relevant to preservation, but not part of preservation itself.

**Example**:

- A *PreservationAction* must produce metadata that is needed by the electronic record management system.

### *3.2.3.5 An in depth analysis of significance constraints*

Because of their central role in digital preservation, this section discusses *Significance-Constraints* in much greater detail than the other *Constraint* categories introduced above.

Custodians of digital content take action when the material that they are responsible for is threatened by, for example, obsolescence or deterioration. At first glance, ideal preservation actions retain every aspect of the original *PreservationObjects* with the highest level of fidelity. However, achieving this goal can be costly, infeasible, and sometimes even undesirable. As a result, custodians must focus their attention on preserving the most significant *Characteristics* of the content, even at the cost of sacrificing less important ones. *Significance-Constraints* that capture these significant *Characteristics* can be considered one specific form of *PreservationGuidingConstraints*. Furthermore, one must verify that the *PreservationActions* one applies actually preserve these *Characteristics*. The concept of significant *Characteristics* has become prominent within the digital preservation community to capture this key goal (Dappert, Farquhar, 2009a).

As is often the case in an emerging field, however, the term significant *Characteristic* has become over-loaded and remains ill-defined. This has some unfortunate consequences. First, communication is hampered, because the term is used in substantially different ways by different authors. Second, based on an extensive analysis of policy and strategy documents related to digital preservation (Clausen, 2007), the current definitions do not actually meet the needs of content custodians. Content custodians need to express priorities, as well as *Constraints* that go beyond the significance of *Properties* and *Values*. Third, implementations based on existing definitions fail to meet the needs of content custodians because they focus too tightly on *Characteristics* of content and format, and do not take account of the context in which *PreservationObjects* exist and in which *PreservationActions* take place.

This section, probes into the meaning of Andrew Wilson's definition of "significant properties" as "the Characteristics of digital objects that must be preserved over time in order to ensure the continued accessibility, usability, and meaning of the objects, and their capacity to be accepted as evidence of what they purport to record." (Wilson,2007).The exploration has led to shifting focus from a priori significance of *Characteristics* in *Files* or file formats to a new model in which stakeholders state *Constraints* expressing significance. In contrast with previous work, this work

- distinguishes *Properties* and *Characteristics*;

- provides a conceptual model, identifies the types of entities which may have *Properties* and *Characteristics*, and unifies the treatment of *Properties* and *Characteristics* across *PreservationObjects*, *PreservationActions*, and their *Environments*;
- clarifies who and what determines significance;
- lists observations about practical uses of *SignificanceConstraints*. They justify why DePICT treats *SignificanceConstraints* as a subtype of *Constraint*;
- clarifies the difference between *SignificanceConstraints*, applicable *Properties* and Representation Information.

DePICT places significance in the hands of stakeholders. The model extends the domain of *SignificanceConstraints* beyond *PreservationObjects* to include *Environments*. The model has consequences for implementations of preservation metadata dictionaries, *Property* registries, and *PreservationServices*. Even though the concept is being discussed within the digital preservation domain, it may also apply to other transformation applications such as rendering accessible versions of digital objects for disabled users.

Modelling Requirements for SignificanceConstraint

All observations regarding *Constraints* in section 3.2.3 also apply to *Significance-Constraints*. Additionally, the following observations need to be considered in a conceptual digital preservation model.

**Modelling Requirement 2.3.11:**

An idea, concept, act, or thing is not inherently significant. A stakeholder attributes significance to something, typically in a context relevant to some purpose or goal. In the digital preservation context, significance is determined by the stakeholders involved in the preservation process. These include the producer of the digital object, the custodian who holds it, and the consumer who will access it. The stakeholder's priorities may be captured as *Constraints* ("business rules") by the custodian, who needs to ensure that *PreservationActions* satisfy them. *Constraints* are an explicit statement of a stakeholder's values. These *Constraints* influence the preservation process, and are often captured in *Policy* documents, such as strategy or business documents. The conceptual model must have a *Constraint* entity for capturing significance explicitly.

There is a notion that *SignificanceConstraints* refer to the intellectual content - the essence of the digital object. In contrast, other *Characteristics* are merely circumstantial, not significant, and can be ignored in *PreservationActions*. Unfortunately, it is not possible to determine out of context which *Properties* reflect content and which reflect circumstance. Consider a number that is formatted with the colour red. In some settings, the colour may be for a visual effect - simply pretty, circumstantial, and insignificant; in another setting, the colour may be to indicate that it is to be understood as a negative number and therefore has a significant semantic impact. This can only be determined by the stakeholder capturing significance explicitly.

**Modelling Requirement 2.3.12:**

A key aspect of the model is that each of the classes *PreservationObject*, *Environment*, and *PreservationAction* illustrated in Figure 33 may have *Properties* and *Characteristics*. It is important to distinguish the types of entity which are characterised. They play different roles during preservation processes and have different applicable *Properties*.

Stakeholders specify *Constraints* on both *PreservationObjects* and *Environments*. Jeff Rothenberg (2000) introduced widely used criteria to evaluate authenticity: content, context, appearance, structure, and behaviour. These are sometimes misinterpreted as exhaustive categories for *SignificanceConstraints* (e.g. Knight, 2008). The consequence is to limit *SignificanceConstraints* to "informational entities" - the logical *PreservationObject* itself - and exclude *Bitstreams*, *Representations*, or *Environments*. Other approaches (Thaller, 2009) limit *SignificanceConstraints* to *Characteristics* of *Bitstreams* or *Representations* since their primary research goal is to evaluate their *Values* automatically from files.

In contrast, the *Characteristics* of *PreservationActions* constrain the context in which *SignificanceConstraints* apply, but are not themselves significant for guiding the *PreservationAction*.

**Modelling Requirement 2.3.13:**

*SignificanceConstraints* are not simple *Property*/*Value* pairs which a stakeholder declares to be significant. The underlying analysis of policy and strategy documents (Dappert, Farquhar, 2009b) shows that stakeholders need to state more complex *Constraints* that can be expressed using a *Constraint* language such as OCL (Warner, Kleppe, 2003). They often need to include specifications such as invariants, pre-conditions and post-conditions. In many cases, a

stakeholder considers *Characteristics* to be significant only when some additional conditions are met - that is, a context is specified.

As a result of Modelling Requirements 2.3.4, 5, 6, 8, 10 and 11, the language that is used to define *SignificanceConstraints* must be able to express relationships other than the simple preservation of a *Value*.

As a result of the above observations *SignificanceConstraints* are defined as:

*Constraints* in a specific context, expressing a combination of *Characteristics* of *PreservationObjects* or *Environments* that must be preserved or attained in order to ensure the continued accessibility, usability, and meaning of *PreservationObjects*, and their capacity to be accepted as evidence of what they purport to record.

Implications for SignificanceConstraint

Using the conceptual model and the definition of *SignificanceConstraint*, one can now investigate some implications of the definition and the relationship of *Significance-Constraints* to related digital preservation concepts.

**Implications of the conceptual model**

The conceptual model suggests the need for developing approaches that allow stakeholders to express *Constraints* with prioritisation and tolerances.

It supports a wide array of preservation activities found in real organisations. *Characteristics* of different entities are used to express *Constraints* for different preservation activities or purposes. For example, bit-*PreservationActions* such as media refresh preserve *Characteristics* at the *File* or *Representation* level such as *fileSize*, *encoding*, or the *numberOfFilesInTheRepresentation*. In contrast, migration actions can be expected to change these *Characteristics*.

*SignificanceConstraints* at the *Representation* level can express *Constraints* associated with the *Representations*' different purposes, such as preservation versus access copies. *Resolution* = "high" and *preservationLevel* = "9" may be *SignificanceCon-straints* of a *Representation* that is aimed at preserving archival quality.

A *SignificanceConstraint* that is considered an inherent *Constraint* of an *Intellec-tualEntity* and does not vary from *Representation* to *Representation* should be

captured on that level. These *Constraints* need to be satisfied by all *PreservationActions* applied to this *IntellectualEntity*. For example the *Constraint semantic-Interpretation* = "negative number" may be declared significant for all representations of a *NumberComponent*. Different *Representations* of the *NumberComponent* can satisfy it by rendering it as a red number, adding a minus sign or surrounding it by parenthesis, but the logical *Constraint* must be satisfied for all of them.

*SignificanceConstraints* of *IntellectualEntities* can model high level policy and strategy *Constraints*, such as legal or fiscal *Constraints* that must be satisfied after any *PreservationAction*.

*SignificanceConstraints* of *Environments* make it possible to express *Constraints* whose aim is preserving the look-and-feel of a *PreservationObject*, since the look-and-feel is determined by the combination of the *PreservationObject* and its *Environment*. These *SignificanceConstraints* support emulation and migration activities equally. Environmental factors can also be external or internal policy factors which permit the expression of policy *Constraints*.

**File formats and properties**

The basic consequence of this analysis is that significance is not inherent in or determined by the file formats of digital objects – but by the needs and *Constraints* of stakeholders in their preservation activities. This enables us to make sense of common preservation activities, such as migration to less expressive file formats. For example, some stakeholders will be satisfied by migration from a *.docx* document to a simple .txt *File* when the original contains only simple *TextComponents* (i.e., no formatting, headers, tables, and so on). A radio station might be satisfied by a migration that only preserves the audio stream of a video object. The analysis also shows why there can be disagreement about the significance of a *Property* between stakeholders. Disagreement reflects different *Constraints* and priorities among stakeholders. For example, the rotational frequency of a shape in a piece of online art may be significant to the artist, but not for many viewers.

The analysis also clarifies the role of archival subsets of *File* formats, such as pdf/a. The well-designed archival format profile will support *Properties* that are of interest to a substantial community of stakeholders and appear in a substantial subset of content in the full file format.

Registries of file formats or content types and the *Properties* that apply to them (e.g. Knight, 2008) are registries of "applicable *Properties*[27]" rather than of "Significant Properties" or *SignificanceConstraints*. A stakeholder is free to indicate that some of the applicable *Properties* are not significant in a certain context. This increases the set of *PreservationActions* that are appropriate. For example, if a Microsoft Word file only contains plain text, then the file's image *Properties* are irrelevant, even though they apply to the file type. This opens up *PreservationActions* with, for example, an RTF format that would otherwise not be possible. Conversely, a stakeholder may indicate preconditions which rule out *PreservationActions* that would have been appropriate considering only the file format's applicable *Properties*. For example, a migration of a .txt file to Abi Word is plausible, based on the applicable *Properties*, but is not permissible, if Abi Word is not supported in the organisational environment.

Under this terminology, it is clear that a *Characteristic* (*Property* / *Value* pair) may be preserved by a *PreservationAction*, but that the abstract *Property* cannot be. It is therefore not sensible to speak about preserving a "significant *Property*."

**SignificanceConstraints and Representation Information**

How do the *SignificanceConstraints* of this conceptual model relate to Representation Information, as defined in OAIS (CCSDS, 2012)? Representation Information is "the information that maps a Data Object into more meaningful concepts. An example is the ASCII definition that describes how a sequence of bits (i.e., a Data Object) is mapped into a symbol."

Representation Information is a set of *Characteristics* describing the *PreservationObject* and its *Environment*. Furthermore, Representation Information is specified for a specific context, namely for a given Designated Community. It will vary for different Designated Communities. Additionally, the purpose of Representation Information is to guarantee the accessibility, usability, and meaning of *PreservationObjects*. All these characteristics of Representation Information agree with the definition of *SignificanceConstraints*. It becomes obvious, that Representation Information is NOT a form of *SignificanceConstraint* when one realises that it does not specify *Characteristics* that need to be preserved or

[27] There are also *Properties* which describe a file format itself rather than the *Objects* that are represented in *Files*. They often appear in stakeholder *Constraints* and enable stakeholders to choose formats that suit their business needs. For example, a custodian might require *Files* to be represented in formats defined by an open standard, or in common use, or with high resilience to degradation damage.

attained, nor does it specify *Constraints* for *PreservationActions*. Representation Information is the set of important *Characteristics* of a Data Object that are needed to make sense of it for a given Designated Community at a given time. It does not specify *Constraints* for transformations over time, and it does not specify *Characteristics* of an acceptable derived Data Object.

A piece of Representation Information, for example, may be the fact that a given Data Object requires a certain software package for its proper rendering. This does not imply that the corresponding Information Object after a migration must use this same software package.

Some pieces of Representation Information may, however, be declared to be significant for preservation purposes. For example, the semantic interpretation of a Data Object, that a given *NumberComponent* is to be interpreted as *bodyWeight*, is likely to be considered significant in most contexts.

### *3.2.4 Policy*

Definition of Policy

Representations that specify *Constraints* that make a stakeholder's values, priorities or goals explicit and influence a *PreservationAction*.

They include oral representations, as well as written representations, in traditional documents, databases, source code, web sites, etc., such as policy, strategy, or business documents, as well as applicable legislation, guidelines, rules, or even a choice of temporary runtime parameters during a *PreservationAction*.

Modelling Requirements for Policy

**Modelling Requirement 2.4.1**

Preservation policies define how to manage digital assets to avert the risk of content loss. They specify, amongst other things, data storage requirements, preservation actions, and responsibilities. A preservation policy ensures the satisfaction of digital preservation goals.

**Modelling Requirement 2.4.2**

Policies are representations which

- may have any institutional scope (corporate, departmental, project related, etc.),

- may have any business focus (policy, strategy, mission, process, etc.),
- provide an input to preservation business processes, such as preservation monitoring or preservation planning. [28]

**Modelling Requirement 2.4.3**

The core of *Policies* are the *Constraints* which are expressed in them. Besides these *Constraints*, however, there are some general aspects which should be contained in *Policies*. DePICT borrows some basics from a model called Strategy Mark-up Language (StratML, nd). It is a basic conceptual model for describing the essential contents of a strategy document. For more information please see section 2.2.4.3.

### *3.2.5 Agents, events and rights*

*Event*, *Agent*, and *Right* are entities that can be modelled in the way they are defined in PREMIS (2012), where *Events* and *Rights* describe *PreservationObjects* and *Agents* refer to either *Events* or *Rights*.

A *PreservationAction* is a special kind of *Event*.

[28] Preservation plans are the output of a preservation planning process and are not considered *Policies*.

# 4 Information exchange model for the digital preservation life-cycle[29]

In order to develop a valid conceptual model for digital preservation, it is important to understand which activities are involved in digital preservation. They are the use cases for which DePICT serves as conceptual model. Using this approach also ensures that the model not only supports the static recording of *Characteristics* and *Events*, but also supports dynamic *PreservationServices*.

Effective digital preservation requires a set of *PreservationServices* that work together to ensure that *PreservationObjects* can be kept accessible and usable for the long-term and that the preservation goals defined in section 1.1.2 are guaranteed. In order to work together, these *PreservationServices* need shared digital preservation metadata, such as descriptions of the *Properties* that *PreservationObjects* or *Environments* may have and descriptions of the *Constraints* that guide digital *PreservationServices*.

Drawing on the practical experience gained in the Planets project (Farquhar, Hockx-Yu, 2007; Planets, nd), this chapter analyses how *PreservationServices* interact and use these metadata. Figure 40 illustrates the roles that *Properties*, *Values*, *Characteristics*, and *Constraint*s (represented by relationship links) play in *PreservationServices* (represented by boxes). By analysing these specific roles, one can derive modelling requirements for key preservation entities, such as *Property*, *Characteristic*, and *Constraint*. The insights gained feed into the development of the conceptual model in appendix 7.1.

[29] The results reported in this section have been published in (Dappert, Farquhar, 2011)

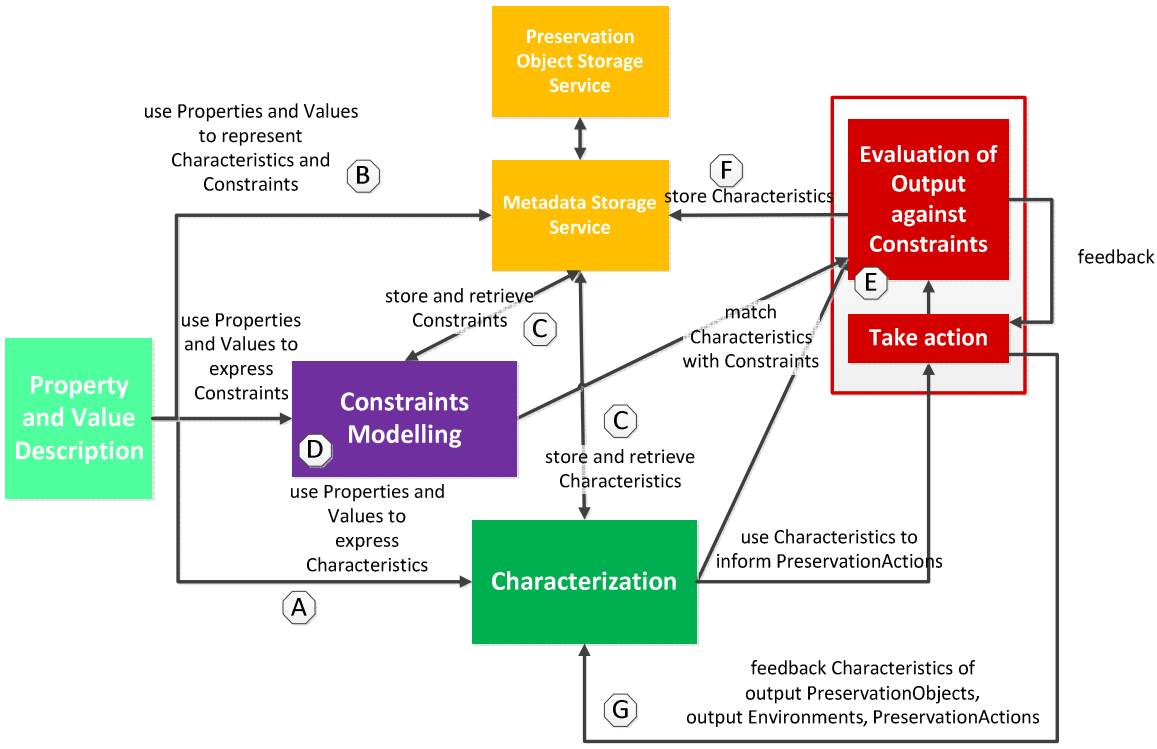



**Figure 40: Interaction of *Properties*, *Characteristics*, and *Constraints***

## 4.1 Property and value description

In order for *PreservationServices* to work together successfully, they need a common definition of *Properties* that ensures interoperability and exchange across not only services, but also systems and institutions. In the preservation community, the definition of digital object *Properties* is currently supported through the following approaches.

Definitions of applicable *Properties* (see Figure 41) can be captured in registries or data dictionaries so that they can be referred to in *PreservationServices*. Alternatively, they can be defined locally for local use in a system.

- File format registries, such as PRONOM (Brown, 2005; TNA, nd), or the Unified Digital Format Registry (UDFR) (nd-a, nd-b), can associate **file formats** with their applicable *Properties*, together with data constraints or a controlled vocabulary.
- The InSPECT project (Knight, 2009) identified *Properties* that apply to **content types**, such as images or emails, rather than to file formats. Other content type specific metadata schemas are, for example, MIX (nd) for image files and textMD (nd) for text files.
- Metadata dictionaries, such as PREMIS (2012) for preservation metadata or PROV Ontology (see section 2.2.4.2) for provenance information, define common preservation metadata elements to describe *Properties* of ***PreservationObjects*** or ***Environments***, together with data constraints or a controlled vocabulary, in a **file format independent** way.
- *Environment* registries, such as TOTEM (Delve, 2011; Delve, Anderson, 2012; TOTEM, nd), National Software Reference Library (NSRL, nd) or the Virtual Resource Description Framework (VRDF) (Kadobayashi, 2010) capture the *Properties* for describing preservation ***Environments*** and their relationships.
- Since related *Properties* are often not immediately comparable, it is useful to develop a *Properties* ontology which captures not only *Properties* of *Preservation-Objects* or *Environments* but also describes them and the relationships between them explicitly. The Planets *Property* Ontology is an example of an ontology that describes file format *Properties*. A subset of it, the XCL ontology, is described in (Thaller et al., 2008; Thaller, 2009; Planets - XCL project, nd).

Definitions of *Property Values* (see Figure 41) can be captured in the above registries or data dictionaries so that they can be referred to in *PreservationServices*. Additionally,

- controlled vocabulary registries, such as the Authorities and Vocabularies service (Library of Congress, nd-b) of the Library of Congress, capture *Properties*' permissible *Values* in order to create a sharable vocabulary for enumerative data constraints which can be used by the above registries or data dictionaries.
- *Characteristics* extraction languages, such as eXtensible Characteristic Extraction Language (XCEL) (Thaller et al., 2008; Thaller, 2009) describe how *Values* for *Properties* can be extracted from files for a given **file format**.
- a *Properties* ontology can capture ways of deriving *Values* in addition to just capturing permissible *Values*. The Planets *Property* Ontology is an example. A subset of it, the XCL ontology, is described in (Planets - XCL project, nd; Thaller et al., 2008; Thaller, 2009).

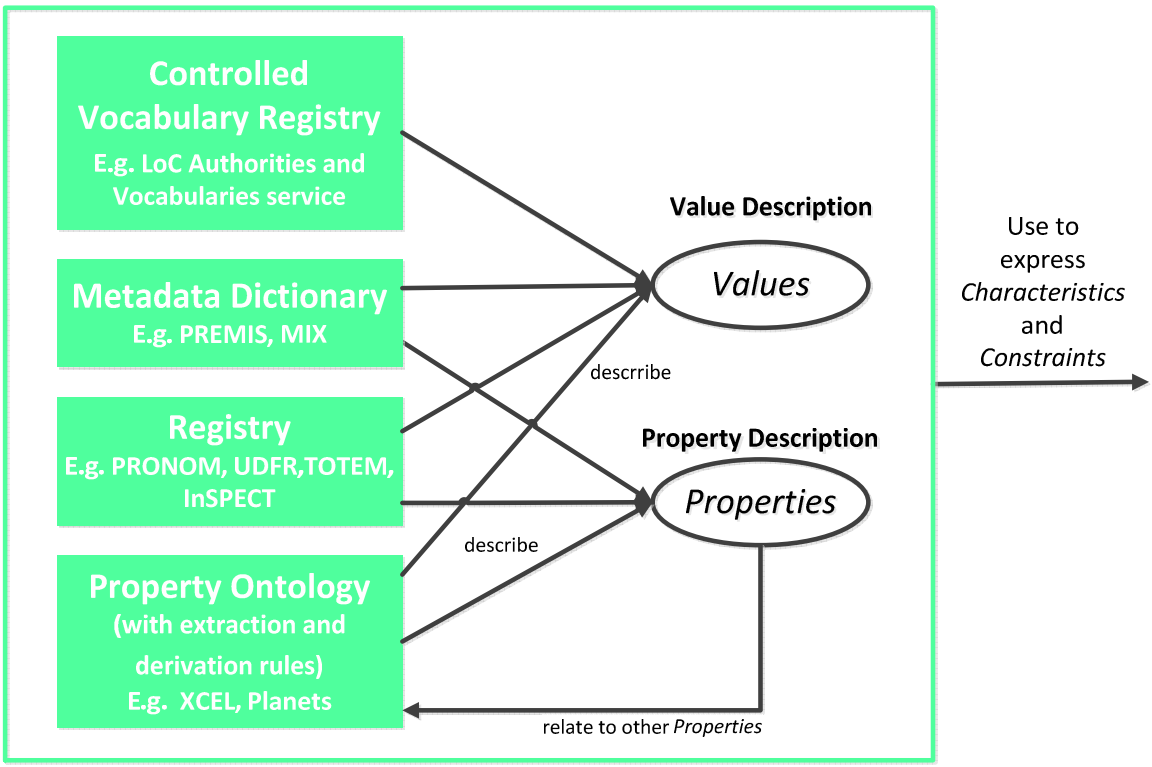



**Figure 41: *Properties*, their descriptions and their permissible *Values* captured by controlled vocabulary registries. LoC = Library of Congress; PREMIS = PREservation Metadata: Implementation Strategies ; MIX = Metadata for Images in XML Standard; UDFR = Unified Digital Format Registry; TOTEM = Trusworthy Online Technical Environment Metadata; InSPECT = Investigating Significant Properties of Electronic Content; XCEL = eXtensible Characterisation Extraction Language; Planets = Preservation and Long-term Access through Networked Services**

Knowledge about *Properties* and their *Values* can be used when

- linking a file format to characterisation services that can determine *Values* for its applicable *Properties* - for example, a service to determine the *fonts* used in a .doc *File* (Figure 40A, Figure 42).
- creating a testbed service that measures the degree to which applicable *Properties* are preserved by *PreservationServices* - for example, measure the degree to which a service preserves *imageWidth* by evaluating it on many objects. In addition to the service *Characteristics* (e.g. *preservesImageWidth* = "no") it can capture the degree to which or under what condition this *Characteristic* holds (Figure 40A, Figure 42).
- enabling metadata storage services to refer to *Properties* and *Values* unambiguously and to ensure interoperability and exchange across institutions and systems (Figure 40B).
- expressing *Constraints* (Figure 40D).
- identifying which *Properties* are shared across file formats and can therefore be preserved by a migration between them (Figure 44).

## 4.2 Characterisation

Characteristics are *Property*/*Value* pairs. They are used to describe *PreservationObjects*, *Environments*, and *PreservationActions*. The following approaches to determining *Characteristics* are in use (Figure 42):

- Characterisation services use file format knowledge to extract ***Property Values*** **from** ***Files*** in order to describe them.
    - **Technical** ***Characteristics*** **of** ***Files*** can be extracted automatically by file format characterisation services, such as, the Journal Storage / Harvard Object Validation Environment (JHOVE and JHOVE2) (JSTOR and the Harvard University Library, nd; Donnelly, 2010; Abrams, Morrissey, Cramer, 2009), the Digital Record Object Identification (DROID) tool associated with the PRONOM database (Brown, 2005; TNA, nd), or any of the other tools analysed in the SCAPE evaluation of characterisation tools (van der Knijff, Wilson, 2011); they use file format knowledge to determine a *File*'s file format, and additionally possibly validate it and describe technical *Properties* of the *File*. They may, for example, determine the dimensions of an image *File*.
    - **File content** ***Characteristics*** **of** ***PreservationObjects*** can be characterised by tools, such as the XCL services (Thaller et al., 2008; Thaller, 2009). They may, for example, determine the *font Properties* of a text string contained in a *File*. This can be used to validate that content *Properties* are maintained after the execution of *PreservationActions*.
- ***Characteristics*** **of** ***PreservationServices*** can be determined experimentally in preservation Testbed services, such as the Planets Testbed service (Aitken et al, 2008). They derive statistics on the performance of *PreservationServices*, such as those performed by a file format migration tool. They determine to what degree those *PreservationServices* preserve *Properties* for representative corpora of digital objects. They, for example, measure the degree to which a *PreservationService* preserves *imageWidth* by evaluating it on many object migrations.
- ***Characteristics*** **of** ***Environments*** can be determined through service dependency analysis. It determines in what way *PreservationObjects* and their *Environments* depend on each other for their proper functioning.

- Computing *Environment* dependencies can be analysed through dependency analysis tools and expressed in metadata languages for their purposes. Hardware dependency can be considered a subset of the larger area of inventory and asset management in common use in systems management. Software dependency can be internal and expressed in metadata definition languages, such as package management systems (e.g. Debian, nd), or can be external and expressed in metadata definition languages such as WSBPEL (2007) or WSDL (W3C, 2001). Tools, such as the IBM Tivoli Endpoint Manager / BigFix (IBM, 2011) create software asset inventories and monitor their usage. Configuration management (ANSI/EIA-649A, 2011) databases hold inventories of all components of the IT infrastructure, including software, hardware, and documentation, and the relationships between them. Application profiling provides a complete overview of a computing infrastructure including virtualised environments. Tools, such as CISCO's Application Profiling Service (Cisco Systems, 2008) "build an in-depth understanding of your application environment by mapping infrastructure interdependencies and application communication flows". Different types of computing dependency analysis are described in the TIMBUS project's deliverable D4.2 (Trezentos et al., 2012).
- *Constraint* dependencies on *Environments* can be analysed
  - manually using modelling software tools making use of modelling languages, such as Archimate (Open Group, 2012) or BPMN (Object Management Group, 2011a). Examples of such tools are Archi (Bolton University, nd) for Archimate and Enterprise Architect (Sparx Systems, nd) or IBM Rational System Architect (IBM, nd), which support the majority of enterprise architecture frameworks;
  - automatically at the business process level; process mining supports the discovery of dependencies through the processing of application event logs. The following link contains a list of process mining tools: http://en.wikipedia.org/wiki/Process_mining#Software_for_process_mining

- Rather than using tools, all *Characteristics* can also be assigned manually.
- *Characteristics* may be stored in metadata storage services or produced on demand (Figure 40C).

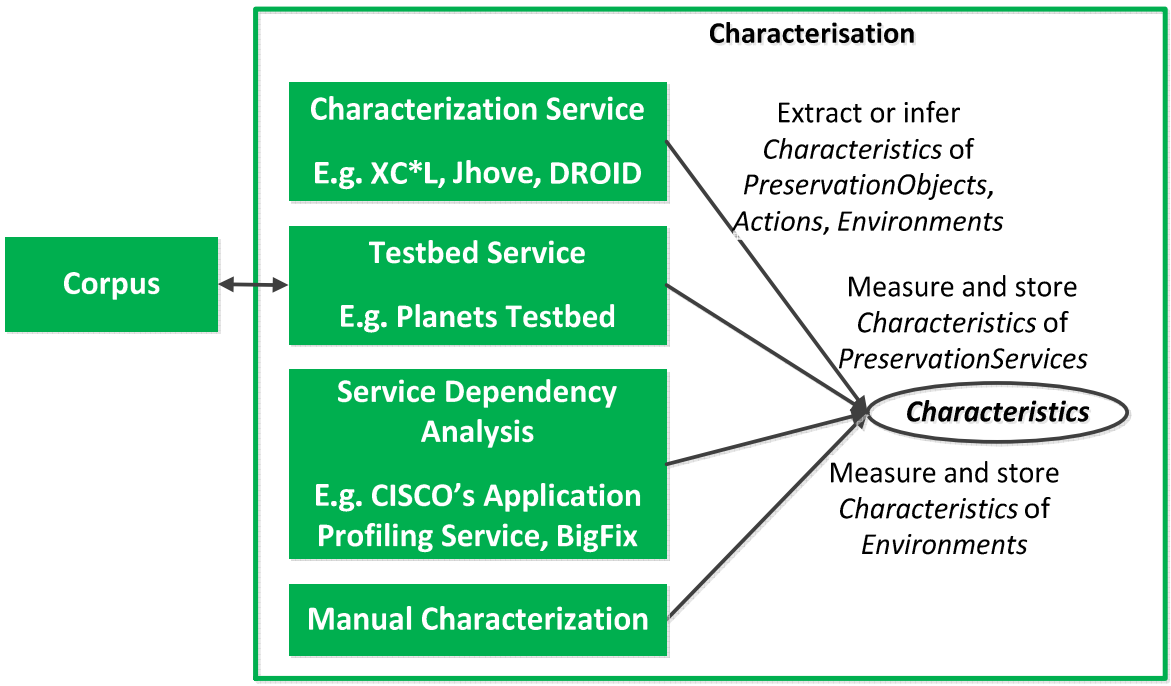



**Figure 42: Characterisation service determination of *PreservationObject*, *PreservationService* and *Environment Characteristics***

## 4.3 Constraint modelling

The process of business modelling results in the formulation of *Constraints*. They are expressed using the *Properties* and controlled vocabulary defined local to the institution or in the standards and approaches discussed in section 4.1 (see Figure 40D). *Constraints* reflect the stakeholders' values, priorities or goals with regard to *PreservationObjects*, *Environments* and *PreservationActions* and guide *PreservationServices* (Figure 43).

- *Constraints* may be captured in *Policies* as defined above in section (3.2.4), that is to say, they may be captured in policy, strategy, business documents, applicable legislation, guidelines, and rules, and also include oral representations, as well as in databases, source code, web sites, or even a choice of temporary runtime parameters during a *PreservationAction*.
- *Constraints* may be preserved in the preservation metadata held in metadata storage services. They document the *Constraints* that have been, or should be, applied to specific *PreservationObjects* (see Figure 40C).
- *Constraints* may be captured in reusable, customisable user profiles which describe the *Constraints* of a default Designated Community.

Little work has gone, so far, into developing digital Preservation *Constraint* modelling tools or languages. The Plato tool (Becker et al., 2008b) uses full-text *Constraints* and captures them in mind-mapping tools. The SCAPE project (Edelstein et al., 2011; SCAPE, nd) is working towards formalising a digital preservation *Constraint* language. Neil Beagrie (Beagrie et al., 2008) has summarised what sort of content should be captured in digital preservation *Policies*.

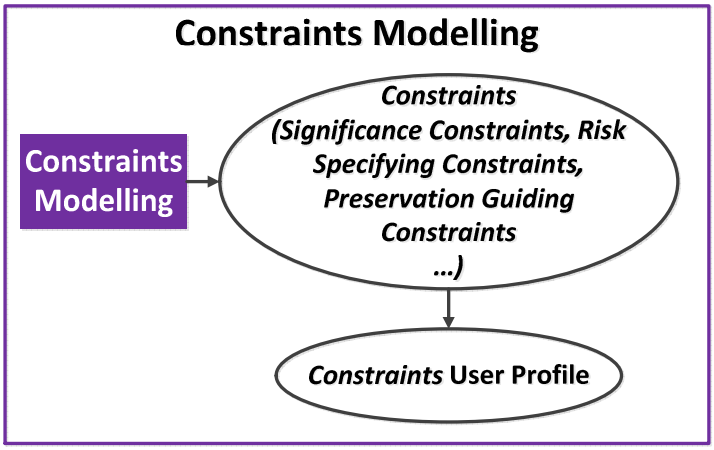



**Figure 43: Formulation of *Constraints* as a result of business modelling**

## *4.4 Characteristics and constraints in metadata storage*

Metadata storage services, such as digital libraries or digital repositories store

- *Characteristics* and *Constraints* of *PreservationObjects* and their *Environments*,
- provenance metadata of *PreservationActions* applied to them,
- the *PreservationObjects* and *Environments* they apply to.

They are expressed using the *Properties* and *Value* vocabulary defined in registries and data dictionaries (see section 4.1) (see Figure 40B, C and F).

## 4.5 Uses of characteristics and constraints

*Characteristics* of and *Constraints* on *PreservationObjects* and *Environments* are used to guide actions by determining which existing *Characteristics* violate or satisfy *Constraints* (see Figure 40E). The primary *PreservationActions* guided by *Constraints* are preservation monitoring services, preservation planning services, and the preservation execution services that perform *TransformationPreservationActions* (see Figure 44).

### 4.5.1 Preservation monitoring

Preservation monitoring services determine whether *RiskSpecifyingConstraints* are violated, indicating that *PreservationRisks* to current and future access to *PreservationObjects* exist (see Figure 44). A preservation monitoring process should trigger the preservation planning process once this happens.

Preservation monitoring services use information about a stakeholder's policies and goals, its infrastructure, its user community, and the external environment in addition to information about the digital objects held within a collection. *PreservationRisks* may be triggered by internal or external change.

Preservation monitoring services goals are to

- identify which parts of the collection present the greatest risks or the greatest opportunities for improvement.

Preservation monitoring services

- need to identify when changes in the *Environment* create new potential *PreservationRisks* to digital content or eliminate previously existing *PreservationRisks*;
- need to, based on this information, update the organisation's business *Constraints* to reflect the applicable *PreservationRisks* for the collections;
- need to determine when one of these applicable *PreservationRisks* for a digital object has arisen;
- and trigger preservation planning.

Tools, such as the Open Planets Foundation's Risk Assessment Tool (RAT), identify the presence of a *PreservationRisk* to *Files* from a set of known risks. Tools, such as the Intelligent

Enterprise risk management (iERM) tool of the TIMBUS project (Edelstein et al., 2011; TIMBUS, nd) help assess the presence of risks to any component of a complete business process, including its `PreservationObjects`. The SCAPE project specifies requirements and the high-level design of a preservation watch system that is currently being developed (Becker at al., 2012).

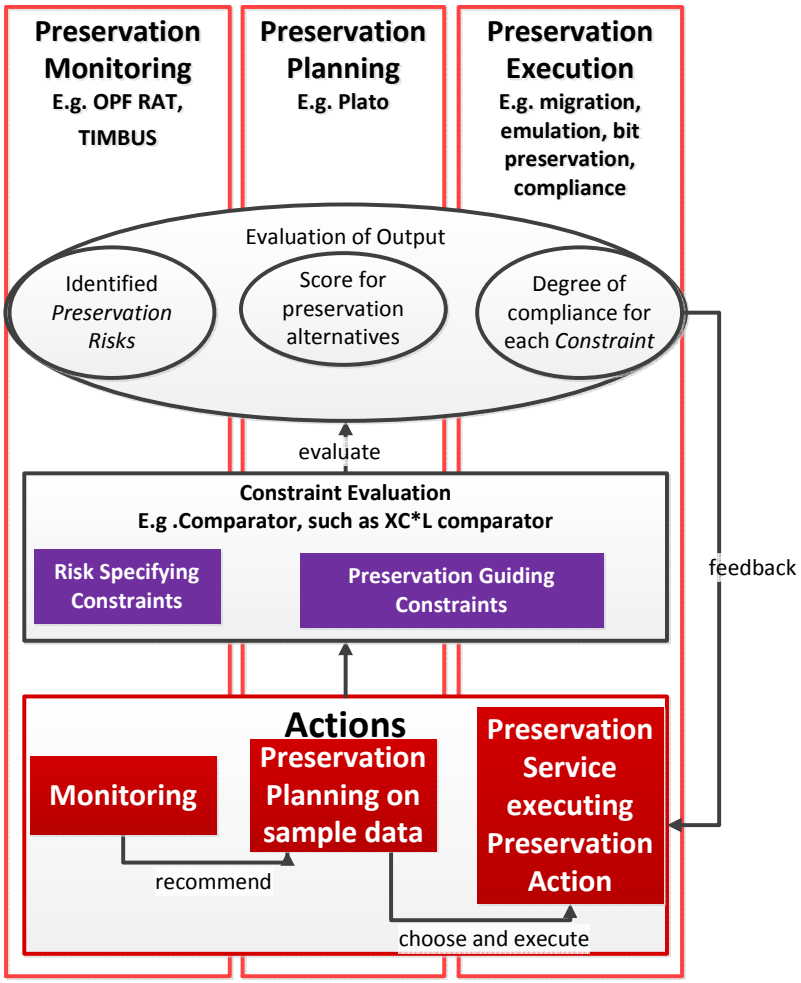



**Figure 44: Uses of `Characteristics` and `Constraints` in `PreservationAction` services**

### 4.5.2 Pre-selection

Optional pre-selection services (not depicted in Figure 44) may provide a prior optimisation step which rules out implausible `PreservationActions`. They analyse `Constraints` to eliminate actions which can from the outset be determined to be violated by `Characteristics` in a given context. Knowledge about the `Characteristics` of `PreservationServices`, which have been obtained in testbed services, is particularly helpful in this step.

### 4.5.3 Preservation planning

Using a sample data set, preservation planning services, such as Plato (Becker et al., 2008b), determine the best choice of preservation execution service to mitigate this identified

*PreservationRisk* (see Figure 44). This is based on an analysis of which *Preservation-GuidingConstraints* are best satisfied by it, and in particular, which *Significance-Constraints* of the sample object set are best preserved. Based on this they recommend the preservation execution service.

Preservation planning goals are to

- identify candidate *TransformationPreservationActions* (alternatives) that could be taken to mitigate the *PreservationRisks*;
- evaluate the candidate *TransformationPreservationActions* to determine their potential costs and benefits;
- weigh the cost/benefit of candidate *TransformationPreservationActions*. The cost may comprise the cost of executing the action, the cost of needed infrastructure for sustaining preservation output, the cost of essential *Characteristics* lost in the *TransformationPreservationAction* (i.e. loss of authenticity) etc. The benefit of the *TransformationPreservationAction* is the benefit of mitigating the risk in terms of the *Value* of the object, the severity of the risk, etc. Obviously these costs and benefits are not necessarily monetary;
- provide justified recommendations for which *TransformationPreservationAction* to execute on which collections.

The result of the preservation planning process is a set of justified prioritised recommendations for *TransformationPreservationActions* that mitigate the *PreservationRisks* presented to *PreservationObjects*.

### *4.5.4 Preservation execution*

The preservation execution service itself performs the chosen *Transformation-PreservationAction* on specific *PreservationObjects* and *Environments* to mitigate the *PreservationRisk* identified by the preservation monitoring service (see Figure 44). There is a wide range of preservation execution services, such as ImageMagick (nd) performing migrations, the KEEP Emulation Framework (KEEP, nd), supporting emulation, or richCopy (Hoffman, 2012) supporting bit preservation.

*TransformationPreservationActions* can create new *PreservationObjects* and *Environments*. Their *Characteristics* may differ from those of the input *PreservationObjects* and *Environments* (Figure 40G).

### *4.5.5 Validation of each action*

Once an action, such as preservation monitoring, preservation planning, or preservation execution, has been executed, it is validated in a *Constraints* evaluation step. The output is either an assessment of the presence and severity of a *PreservationRisk*, or a measure of the degree of compliance of an action with the set of *Constraints* (Figure 40E, Figure 44).

*Constraint* evaluators should determine the degree to which *Characteristics* of the *PreservationObjects*, *PreservationActions*, and *Environments* before, during, and after actions comply with *Constraints*. For example the eXtensible Characteristic Definition Language (XCDL) comparator (Thaller et al., 2008; Thaller, 2009) compares *Characteristics* of *File PreservationObjects* before and after migration. A migration preservation execution service, such as ImageMagick (nd) may be evaluated for their quality based on comparisons of the preservation of the *Characteristics* of the input and output *PreservationObject*. But it is also necessary to consider other *Constraints*, such as the satisfaction of statutory rules or the cost of action. On other types of preservation methodologies, such as emulation, *Constraints* are validated by comparing the input and output *PresevationObject* performances.

### *4.5.6 Provenance*

*Constraints* that are used by these services can serve as explicit provenance information. A metadata storage service should document the provenance of a repository's *Preservation-Objects* (Figure 40F). For each *PreservationObject*, it should record the *Preservation-Actions* that impacted it, the set of *Constraints* that applied at the time and the mechanisms used to validate the *Constraints* during the *PreservationAction*. This information can clarify what happened at that time and why. It can also store the *PreservationObject*'s degree of compliance with respect to each *Constraint*, especially its *Significance-Constraints*. If a *Characteristic* of an input *PreservationObject* is not specified by a *Constraint* it can be lost during a *PreservationAction* without this fact being explicitly noticed; it is then not possible to record their loss. Practically it would be expensive to formalise every applicable *Constraint* or to list them exhaustively, but *SignificanceConstraints* should be documented.

### *4.5.7 Preservation services interactions*

The preservation community is developing a set of tools that delivers digital *Preservation-Services*, such as the tools supported by the Open Planets Foundation (OPF, nd). *Properties* of digital objects play a central role in how these digital *PreservationServices* co-operate. All key *PreservationServices* are linked via a common understanding of the *Properties* which can be used to capture the description of a digital object in a repository's care.

Unfortunately, as observed above (see section 3.1.4.1), different services tend to express *Properties* at different levels. There is, for example, a gap between the *Properties* extracted by typical tools and the *Properties* that stakeholders use to express their preservation *Constraints*. It also has been observed that *Values* for *Properties* may be obtained in different ways; this may result in different observed *Values*. Additionally, inherent differences between file formats make the comparison of some *Properties* difficult.

It is, therefore, important to analyse the actions and interactions of *PreservationServices* to determine what sorts of *Properties* are expressed and exchanged, in order to determine where possible misalignments of definition may occur. They may happen in the following situations.

- **Preservation planning and preservation execution services**

Stakeholders specify *SignificanceConstraints* of their *PreservationObjects* and *Environments* that need to be preserved (or obtained) through a *Transformation-PreservationAction*. Preservation planning and preservation execution services need to determine reliably whether these *SignificanceConstraints* have been satisfied. They request the *Values* for the *Properties* mentioned in the *SignificanceConstraints* from the preservation characterisation service. The characterisation service is supposed to deliver the *Values* for these *Properties* in the requested form or in a form that can be converted to the requested one. The preservation planning service additionally requests *Characteristics* that describe the *TransformationPreservationAction* tools' performance from the testbed service in order to select tools that suit the sample data. These also need to align with the *Properties* expressed in the *SignificanceConstraints*.

- **Preservation monitoring**

Policy documents can specify which *Characteristics* of *PreservationObjects* and their *Environments* manifest a *PreservationRisk*. In order to determine whether a *PreservationObject* is at risk the monitoring service requests the object's *Characteristics* from the characterisation service. The *Properties* used by the two services need to align.

- **Testbed experimentation**

During a testbed experiment, a preservation execution service is tested on a set of *PreservationObjects*, called a corpus. During the test, derivative objects are created whose *Property Values* are compared to the *Property Values* of the original objects. The results of this comparison describe the behaviour of a preservation execution service based on the degree to which the service preserves the *Properties*' *Values*. There are two possible clashes. Firstly, this result is only meaningful if the testbed tests for a set of *Properties* that are relevant to the users whose *Constraints* are captured by preservation planning services. Therefore the *Properties* used in preservation planning and those tested in the testbed should align. Secondly, the testbed needs to obtain *Values* of the measured *Property* from preservation characterisation services and their *Properties* need to align.

Secondly, the testbed needs to aggregate test results that describe tool characteristics (rather than object characteristics) in a way that is most meaningful to their users and write them to a registry ready for use. Preservation planning services weigh those service *Characteristics* to determine the optimal service for the users' specific preservation needs. The *Properties* used by both need to align.

- **Corpus design**

A corpus is a set of digital objects with known *Characteristics* for use in experiments. In order to compile benchmark corpora on which one can run testbed experiments in a representative way, one has to have an understanding of the applicable and relevant *Properties*. Testbed results are meaningful to preservation planning services only if they are derived on a corpus of digital objects that reflects real life applications and contains instances of all *Properties* that are relevant to users. It is, therefore, important that a corpus covers all *Properties* that might be expressed by users in *SignificanceConstraints*.

- **Enhancement of preservation execution service tools**

Developers of a migration tool must ensure that a digital object after migration with this tool has the same *Properties* as the digital object before migration. One way to achieve this would be to specify which *Property* of the source format is to be transformed into which *Property* of the target format. A test migration might then be carried out using sample *Files* the results of which might be tested to determine whether the assessment of *Property* relationships was accurate, and whether the migration tool maintained the *Properties* faithfully. The

*Properties* of the source and target *File* format need to align. Similar considerations apply when assessing the fidelity of emulation.

Another approach might be to ask human subjects to assess the degree of conformance of the target to the source object. The *Properties* that the human subjects apply are not necessarily the *Properties* which were defined by the tool developers. In this case corrections of the *Property* relationships and of the tool are necessary.

# 5 Validation and valuation

The DePICT model was developed as original research while the author was working on the Planets project (Farquhar, Hockx-Yu, 2007; Planets, nd), the SCAPE project (Edelstein et al., 2011; SCAPE, nd) and the TIMBUS project (Edelstein et al., 2011; TIMBUS, nd). These large scale EU co-funded projects presented an ideal testbed for examining concepts, properties and requirements applied in digital preservation methodologies and tools, and to investigate their information needs and information exchange. The three projects had different foci: interacting preservation services and tools covering the whole of the business-cycle; scalable solutions for large collections or for collections consisting of large, complex or heterogeneous objects; and preservation of processes and third-party dependencies with some specialisation on legal issues affecting digital preservation. Being able to study the field under these different perspectives enriched the model and ensured thorough coverage. At the same time the author was employed by the British Library and the Digital Preservation Coalition, which permitted ready access to content owning experts and practitioners who were willing to test the model and to be interviewed about their collections, their decision making approaches and constraints applying to their digital preservation practice. It also permitted access to large-scale digital collections to understand the properties of a large variety of different content-types. It finally also permitted an appreciation for real-life business processes and pragmatic business needs. The author also served on the PREMIS[30] Editorial Committee that strives to provide a data dictionary as *de facto* metadata standard, with which the digital preservation community can capture its digital preservation metadata needs. Intimate familiarity with the dictionary resulted in the author's awareness of short-comings of the current solution, her ability to influence changes to the *de facto* standard, and the ability to closely interact with the user community to understand user needs in practice.

DePICT is theoretically and empirically founded. Techniques used for information gathering included literature analysis, document analysis, software tool and services design and planning meetings, software tool and services analysis, personal interviews, one-to-one and group discussions, publications, presentations and discussions at conferences and EU Reviews, and a workshop with other EU project work-packages.

There were 3 major iterations; after each iteration, the model was reported to the scientific community and externally evaluated. Within each iteration the model was incrementally improved whenever a new information source was investigated. Table 4 gives an overview over the methodologies employed. At this point the model has reached a stable state.

[30] http://www.loc.gov/standards/premis/premis-editorial-committee.html

In the first iteration a first draft model was created and refined through literature and document analysis and expert interviews.

- The first iteration was started off with the creation of a preliminary model from first principles. It was determined what scope, context and functions in digital preservation should be addressed, and what concepts should be present in a model to support them. From this analysis a first draft model was created.
- The first round of iterative refinement and improvement of the model was then based on analysis of how digital preservation practitioners – implicitly or explicitly – define and materialise their commitment and effort to digital preservation in their de facto solutions in use. Relevant concepts and vocabulary from the material was extracted to populate the model and a list of example constraints was compiled. This step resulted in a first list of requirements that the DePICT model would have to satisfy.
    - Organisations involved in digital preservation have created documents describing their policies, strategies, work-flows, plans, and goals to provide guidance. They capture many of the concepts that are seen to be important by decision makers. The actual preservation guiding documents, such as policy and strategy documents (called Policies in the model) from archives, national libraries, and data centres were analysed for their content, such as the digital preservation policies of the National Archives of Australia (2005, 2011), the Florida Digital Archive (2006, 2011), the Hampshire Record Office (2005), the British Library (2007) and the UK Data Archive (UKDA) (2005, 2011).
    - Organisations involved in digital preservation also have skilled staff who are aware of sometimes unwritten considerations. Decision makers from libraries, archives, and data centres that are actively engaged in digital preservation were interviewed (Dappert et al. 2008) to determine factors that influence their preservation decisions.
    - Additionally, a list was compiled of observed constraints that guide digital preservation actions and that were used by stakeholders in policy and strategy documents and mentioned in expert interviews. This list was used to experiment to what degree it was possible to express and process these constraints in a fully machine-interpretable way and to conduct automated reasoning with them. Figure 31 and Figure 38 show examples of possible machine-interpretable formulations. This exercise highlighted the limits of automation. Many

characteristics and constraints would require considerable effort to formalise in a machine-interpretable form – currently well beyond the resources available to the main stakeholders. Even in these cases, however, the model provides value, guidance for analysis of digital preservation situations and a framework for communication. This phase of the work also helped to refine the model, identify the relevant entities and clarify the relationships between them.

- To complement this, an analysis of theoretical literature on digital preservation conceptual topics and abstract definitions of preservation policies and preservation strategies was performed. This established further requirements for the model. The scientific literature was examined for definitions of terminology, concepts and content related to preservation policy, such as the definitions used by the American Library Association (ALA) (PARS, 2007), the concepts appearing in the Audit Checklist for Certifying Digital Repositories of the Center for Research Libraries (CRL)/Online Computer Library Center (OCLC) (2007), the Cornell University Library's Digital Preservation Management Workshops and Tutorial terminology pages (Cornell University Library, ICPSR, nd), concepts used in the Electronic Research Preservation and Access Network (ERPANET) Policy Tool (2003), the Joint Information Systems Committee (JISC) Briefing paper (JISC, 2006) and Solinet (2005).
- The DePICT model was refined by contrasting the resulting model with existing conceptual models of digital preservation, such as PREMIS (2012), OAIS (CCSDS, 2002), the SDB model in use by the Tessella company (Tessella, nd), and other work described in the related research chapter 2. Wherever possible the terminology and the model was aligned with PREMIS (2012), as the leading digital preservation data dictionary, and OAIS (CCSDS, 2012), the accepted framework for archival information systems. DePICT was also compared against existing conceptualisations of the domain in order to discover the gaps that currently don't meet user requirements and to bring out in what way DePICT could improve upon existing models to meet user requirements. The results of this latter work are reported in chapter 2.

Results from this iteration were reported in Planets report PP2-D2 (Dappert, Ballaux, Mayr, van Bussel, 2008) and in the peer-reviewed publication (Dappert, 2009).

In its second iteration, theoretical and practical experiences gained in various EU projects and elsewhere were used to continuously improve existing models and validate the resulting model: against concrete tools and services; by analysing the collaboration and information exchange of

services; and by applying the model to find solutions for two open conceptual problems in the domain.

- Close cooperation with research staff who developed software tools to support a variety of digital preservation services contributed to a deeper understanding of the domain. In order to ensure DePICT's practical applicability for preservation services, the model's concepts and vocabulary were validated through application in or alignment against, in particular, the broad array of preservation services implemented by the Planets work-packages (2006 -2010) (Farquhar, Hockx-Yu, 2007). The project has been recognised for its impact on the preservation landscape. It was short-listed for the Digital Preservation Award in 2010, and, in 2012, was awarded the Digital Preservation Award for Research and Innovation. A selection of its software tools are now being maintained and further developed under the auspices of the Open Planets Foundation. DePICT was also validated against some work coming out of the KEEP (2009 – 2013) (KEEP, nd), TIMBUS (2011 – 2014) (Edelstein et al., 2011; TIMBUS, nd) and SCAPE (2011-2015) (Edelstein et al., 2011; SCAPE, nd) projects.
    - Analysis of tools and services determined which information on which concepts is used and produced by them. It was discussed where DePICT might need to be changed to accommodate these applications, or where it exceeds the local usage. A gap analysis was performed on which of their aspects are not supported by existing conceptual models.
    - Requirement trees used in the preservation planning tool Plato (Becker et al., 2008b) are mind-map representations of the constraints that need to be considered to perform preservation planning. Sample trees for the use cases collected through Plato were implemented using DePICT. This exercise illustrated the extent of expressiveness required for automatic reasoning that could not be accommodated by the Plato tool, but validated the basic concepts and relationship of the DePICT model.
    - The British Library approached the author and organised sessions in which the model was applied to the design of the planned metadata management component of their Digital Library System. The analyst who applied the DePICT model found it to be helpful. He felt that it particularly provided support for ensuring coverage of digital preservation aspects that needed to be considered.

- Two studies on functions, interactions and information exchange were conducted in order to ensure that all needed digital preservation functionality was covered by DePICT and that the model held up to dynamic use.
    - The analysis of the interaction of the preservation services implemented by the Planets project resulted in a unified functional model that illustrates what information is created, used and exchanged between services. It relates this process back to the DePICT model. This work is described in chapter 4 and (Dappert, Farquhar, 2009a).
    - A workshop with all Planets work-packages was organised by Barbara Sierman. It studied the functional models for digital preservation in OAIS and Planets and resulted in her report (Sierman, 2009) on the Planets preservation planning process model and a gap analysis of the OAIS model. This workshop was also used to ensure that the DePICT information exchange model that is presented in section 4 was aligned in functionality, terminology and coverage.
- As DePICT was reaching a more mature state it was used to clarify two particularly interesting issues that had previously been raised in the digital preservation community. They could now be analysed in depth as the entities in question were soundly embedded in the coherent DePICT context model.
    - The discussion in section 3.1.4.1 deals with the problem that two major tools in the Planets project had in communicating with each other. The characterisation tool extracted characteristics on a file level, the planning tool expressed constraints on a user requirements level. There was a conceptual gap between those representations. Section 3.1.4.1 investigates from where this gap derived and how it could be overcome by use of ontological modelling (Dappert, 2010).
    - It was observed that “significant properties” (or similarly named concepts) and “representation information”, both key concepts in the domain, were used in incompatible ways throughout the community. A conference held in 2008 (Hockx-Yu, Knight, 2008) illustrated the lack of common understanding and prompted the effort in this thesis to clarify the concepts by tying them into the DePICT model. The research is reported in section 3.2.3.5. The corresponding publication had a substantial impact on the community’s understanding of these issues. An excerpt from Yeo (2010) illustrates one of the aspects in which this work has been influential.

> *"However, a presentation by Angela Dappert and Adam Farquhar at the ECDL conference in 2009 reminded digital preservation specialists that significance is not universal and must always depend on the varying needs of 'stakeholders'; and the final outputs of the InSPECT project, which became publicly available while this article was being peer-reviewed for Archival Science, adopted a very different stance from the project's earlier publications, recommending a detailed evaluation of 'stakeholder functional requirements' as well as analysis of the properties of digital objects (Dappert and Farquhar 2009b; InSPECT 2009[31])."*

Results from this iteration were reported in Planets report PP2-D3 (Dappert, 2009) and peer-reviewed publications (Dappert, Farquhar, 2009a; Dappert, Farquhar, 2009b; Dappert, 2010; Dappert, Farquhar, 2011).

In the third iteration, the relationship of DePICT to existing standards was tested and DePICT was applied to improve them.

- The relationship of DePICT to the risk-management standard ISO 31000 (ISO, 2009) was analysed. The services identified in chapter 4 and depicted in Figure 40 explicitly represent the ISO 31000 risk management process of monitoring, establishing the context, risk assessment, mitigation planning, and risk treatment (Dappert, 2011). Risk management was one of DePICT's driving motivations. The information exchange study is embedded into this risk management life-cycle.
- The relationship to the PREMIS model (PREMIS, 2012) was studied in depth. DePICT draws from the PREMIS data dictionary in order to align itself with community-standards as much as possible, but also feeds into it, since the author serves on the Editorial Committee. DePICT was used to develop concrete change proposals to the PREMIS data dictionary to test for practical implementability of DepICT ideas and to apply DePICT's insights for the benefit of the community. An analysis of how PREMIS could be improved through aspects of DePICT is provided in section 2.2.4.1. Based on the work presented in DePICT, the PREMIS Editorial Committee will make IntellectualEntities a sub-class of Objects in version 3.0. The need has been independently validated, for example, by a request from the PREMIS Implementers Group in autumn 2012 to provide the ability to attach Events to IntellectualEntities. This will be automatically accommodated by this agreed proposal.
- The model was tested against extended representation needs for capturing complex computing environments and process descriptions within the TIMBUS project (Edelstein et al., 2011; TIMBUS, nd) and within the PREMIS data dictionary. A further planned

[31] Citations from Yeo's paper are adapted to the citations in this thesis.

modification for PREMIS version 3 is the improved handling of Environments. The author has been playing a leading role in the Working Group that is preparing the proposal (Dappert, Peyrard, Delve, Chou, 2012) planned to be released in 2013. A detailed analysis of the limitations of the current PREMIS solution for capturing information about computing environments is provided in section 2.2.5.2.1. The DePICT solution will remove these limitations, thereby improving the prospects for international interoperability, where renewed interest in finding emulation solutions, and in creating technical registries, requires shared models. In order to validate the applicability and appropriateness of the proposed data dictionary changes, the working group has tested use cases from the TIMBUS project, which aims at preserving business and scientific processes so that they can be redeployed at a later point, (Dappert, Peyrard, Delve, Chou, 2012) and from the wider community. DePICT draws from and feeds into context and constraints modelling as executed in the TIMBUS project .

- Engagement with the PREMIS user community to determine unmet needs has provided further information on, often very detailed information needs. Requirements discussed had either already been met by the DePICT model or have let to its improvement.

The results from this iteration were reported in an invited conference presentation (Dappert, 2011), at the PREMIS user group meeting 2012 (Dappert, 2012) and in peer-reviewed publications (Dappert, Enders, 2010; Dappert, Peyrard, Delve, Chou, 2012).

The model has arrived at a stable version which satisfies the requirements emerging from the investigated work. Once the stable state was reached, a final, formal conceptual model expressed in UML was created from the collected model requirements. A corresponding appropriate machine-interpretable model as an XML schema was implemented. Further proof of concept will now require use of its features in more implemented systems.

The model and the research reported in this thesis are having an impact on the key standard in the field, as insights gained during the model development have fed into the improvements to the PREMIS data dictionary from version 2.1 to the expected 3.0. The model is already being integrated into the work of institutions. There is the possibility of significant impact from this model.

# 6 Conclusion

## 6.1 Scope of the contribution

Section 1.2 of this thesis introduced the research goal of creating a conceptual model for the field of digital preservation that expresses its core concepts and constraints - DePICT (**D**igital **P**reservat**I**on **C**onceptualisa**T**ion). Section 1.2.1 analysed the gaps in the existing approaches which prevent their end-to-end life-cycle applicability. In chapters 2 and 3 the key entities in the field were analysed in detail and key requirements for their use and information flow were derived. Appendix 7.1 presented a compilation of this informal conceptual model into a formal UML model of the domain and delivered one (possible) serialisation of this model in XML as the basis for automated preservation services.

In particular, the research outputs of this thesis are

- a conceptual model of the digital preservation domain, based on domain requirements (see chapter 3).
- an UML implementation of the conceptual model (see appendix 7.1) which can be reused by digital preservation researchers and developers.
- a machine interpretable implementation of the conceptual model (see appendix 7.2) that can be used by preservation services.
- an example scenario (see chapter 7.3).
- a top-level vocabulary for the concepts in the model (see chapter 3). DePICT develops a common top-level structure, and provides guidance to stakeholders on how to use and extend the conceptual model. The top-level vocabulary for each entity can be extended by specialist vocabulary as needed.
- an analysis of the role of digital object properties and characteristics (see section 3.1.4). Interesting relationships between properties of digital preservation objects and their environments occur in the digital preservation process that are not straight-forward to resolve. This thesis investigates how a property ontology can be used to model them explicitly in order to overcome possible misalignments.
- an analysis of constraints that guide digital preservation processes (see section 3.2.3.2.2). In particular, this thesis considers `SignificanceConstraints` one specific form of preservation guiding constraint. It examines the concept of “significance” of the

properties of preservation objects in digital preservation, which determines which properties must be preserved over the long-term. It presents a new model that places significance in the hands of stakeholders. The model also extends the domain of *SignificanceConstraints* beyond digital objects to include environments.

This analysis applies to the digital preservation domain, but may apply to other transformation applications, such as rendering accessible versions of digital objects for disabled users.

- an analysis of how preservation services interact and use preservation metadata dynamically, and of how properties, characteristics and constraints affect the interaction of digital preservation services (see chapter 4).

DePICT represents a significant contribution to the field:

- It can be shared by institutions and software applications to improve the exchange and the interoperability of data, metadata and software.
- It can provide a standard which can serve as a convenient starting point for creating individualised models for an institution, saving them time and helping avoid errors. This holds true even if the institution does not require a machine-interpretable specification. Institutions can reuse the high-level specific vocabulary for expressing their own policies and strategies and describing their processes.
- It can be used to describe preservation metadata for individual institutions, possibly, but not necessarily, in a machine-interpretable form, that guide preservation actions. This, in turn, enables preservation services and decision support to be based on organisational policy and strategy constraints.
- It adds to the scientific understanding of digital preservation.

This thesis makes a contribution towards protecting the substantial investments which have been made into the creation of digital assets. It has ramifications in a wide array of sectors:

- memory institutions,
- higher education, and
- industries, which are rich in digital information that needs to be preserved in the longer term.

It provides a conceptual model

- for scholars who conduct research on digital preservation,
- for preservation experts at institutions who actively preserve their digital collections,
- for digital content owners who specify policies and strategies for their collections,
- for digital preservation tool developers.

Such a model supports implementations of

- digital object repositories,
- preservation metadata dictionaries,
- digital format, technical environment and property registries, and
- digital data management and preservation services.

## *6.2 Research contributions*

Chapter 2 introduces existing conceptualisations of the preservation domain and examines their short-comings; section 1.2.1 summarizes the observed short-comings into a list of 6 main modelling gaps. The DePICT conceptual model improves upon them, and overcomes these short-comings in the following way:

It is suitable for

- modelling a very wide range of preservation services, such as risk monitoring; determining characteristics of objects and environments including tools; comparison of characteristics to determine authenticity; evaluation of candidate preservation actions; and evaluation and validation of executed preservation actions. This is demonstrated in detail in chapter 4.
- modelling a very wide range of entities from logical to physical entities (intellectual entities, representations and bitstreams), including preservation actions and environments, as discussed in section 3.1.1.
- modelling technical as well as organisational properties, incorporating all relevant organisational characteristics and strategic directions. This is achieved by introducing the

implementation-independent and technology-independent high-level entities of properties and constraints, rather than by elaborating specific lists of technical properties. It is also achieved by suggesting an extensible top-level vocabulary for their sub-classes that is not restricted to technical aspects. It is furthermore supported by the fact that the model has abstract and physical preservation object types (e.g. intellectual entity and bitstream) which encourages a view that comprises all relevant levels of abstraction. These approaches explicitly direct attention to the global context.

- modelling a very wide range of preservation methodologies. This includes traditional actions, such as emulation, virtualisation, migration, recreation and bit-preservation. But it also includes more abstract but equally important actions that update metadata or features of digital materials so that they comply with current legal or user requirements. Modelling a very wide range of preservation actions is enabled by incorporating a non-technical view. It is also enabled by modelling preservation objects and environments as closely interlinked entities that assume the same relationships to other entities in the model. In general, preservation actions affect the whole rendering and execution stack. The content, its representation, its implementation and its storage are mutually dependent and affect each other. Taking this view is necessary whenever preservation actions apply to more than the isolated preservation object files.

- basing preservation actions on risk management by lining up preservation execution services against the preservation risks they mitigate. Analysis showed that decision-making experts consider preservation risks as drivers for preservation actions in their daily work and capture them explicitly in their policy and strategy documents. Previous models have failed to capture this and often modelled actions as performed and evaluated against an absolute standard.

  Preservation risks can be captured as constraints, which, in turn, are expressed through the properties of preservation objects and environments. These shared properties link preservation risks and preservation actions together in a logical way. Preservation risks and other constraint statements in the digital life-cycle, such as the ones that guide preservation action choices, or define authenticity of outputs after performing preservation actions, are evaluated in the same way. This is shown in chapter 4, and in Figure 40E and Figure 44.

Knowledge about preservation risk is also vital provenance information that explains to future stakeholders why certain actions were chosen. It must be preserved as part of the life-cycle information. Current models don't enable capturing this information.

- covering the full life cycle of digital information objects from the moment of creation to their deletion or "tombstoning"[32]. This is demonstrated in detail in chapter 4. Figure 40 shows the cycle of creating values for properties of preservation objects, environments and preservation actions, evaluating them against constraints, and using this evaluation in order to trigger actions. Execution of these actions results in the creation of new property values based on the action outputs, which starts the cycle once more in a changing world. Not surprisingly, this cycle mimics exactly the iterative approach defined in the risk management standard ISO 31000 (ISO, 2009).

The DePICT model is the first comprehensive model of the digital preservation domain covering all of these aspects. All other models have been partial models. As a comprehensive model it helps defining digital preservation as a coherent discipline and supports the development of effective, collaborative end-to-end service.

[32] where the object is deleted but some metadata and its persistent identifier are maintained for documentation purposes and so that other objects can still refer to the deleted object

## 6.3 Further work

There are important avenues for future work. Further significant contributions can be made in the areas of embedding the model in the practical problem solving process, model extensions, extending the model to connect with registry modelling, and distinguishing the varying preservation needs of the parts of the complete rendering stack.

### 6.3.1 Assisting practical application

The DePICT model has been created based on theoretical analyses of practical applications. It is now important to test and validate it in newly developing situations and edge cases. In a first, important step, aspects of it are being integrated into the PREMIS *de facto* standard. This enables access to a wide user base that can then provide feedback for future improvements.

But, models are notoriously hard to apply for inexperienced users and assistance is required for making them usable. To support implementation, guidelines and best practice recommendations need to be developed. Even though the model is rather slim, there are several justifiable choices for how to implement specific instances, for example, which entities to choose for which real life object and how to relate them. As a further help, reusable templates could be developed for very common instances of preservation systems. Such exemplars for common decision situations tend to be very helpful.

It is also desirable to enable a modelling approach in which digital and non-digital preservation objects can be handled together and as uniformly as possible and integrated into one management workflow where appropriate. It will be required, in practice, to investigate whether and where exactly the DePICT model differs from approaches in the non-digital world, and where it overlaps.

### 6.3.2 Model extensions

Developing granular expert vocabulary or ontologies for the core entities has, from the outset, not been in the scope of this thesis, and will be left to stakeholders of specific domains. They will provide important extensions for particular application areas. DePICT is a high-level conceptual model that applies to all digital preservation scenarios. It defines entities that apply universally. DePICT's entities can be made more specific by creating sub-classes and by refining the entity space hierarchically. The DePICT description in section 3 illustrates for each entity in a "Vocabulary for <Entity> sub-classes" section how it can be extended through common refinement categories but does not define them as part of the conceptual model.

As stated in section 2.2.3, because of the breadth of metadata needed to support the full range of digital preservation goals and the variety of scenarios in which digital preservation is applied, it does not make sense to create one monolithic data dictionary to be used by all and to apply to all situations. Many years of expertise and effort have already gone into specifying metadata dictionaries or implementation specifications for subsets of the four metadata categories listed in section 2.2.2 that are also used to support functions outside digital preservation. There is no point in trying to reproduce or outdo these efforts. Additionally, it is not possible to define one set of metadata that applies equally to all content types or organisation types. Archival records, manuscripts, and library records, for example, require different descriptive metadata; images, text-based documents, and software source code require different technical metadata. Because of this, a number of metadata definition efforts have evolved, both in a content type- or organisation type-specific space and a preservation function space. Different metadata specifications can be combined by using a container metadata schema, such as METS (METS, nd) that defines metadata categories, and relationship and identifier mechanisms through which descriptions in different specifications can link to each other.

Figure 9 illustrates this in a very simplified way. Several of these initiatives have reached the status of a standard or are *de facto* standards.

In order to be flexible and apply to a wide range of contexts, DePICT, like other general preservation metadata and metadata container specifications tries to avoid content and organisation specific semantics. To add specificity, general metadata specifications include extension methods to support content or organization specific metadata. These more specific metadata specifications provide complete sets of properties to describe specific contexts. They provide improved interoperability between independent organizations which share identical contexts; but they may be overly specific and exclude possible other uses. This can stimulate the development of multiple, incompatible metadata solutions to accommodate minor variations in requirements. It is difficult to strike the right balance between generality and specificity. Nonetheless, reusable frameworks with well-defined extension points that allow for specific community agreed schemas have been a major advance.

When combining different metadata specifications or when embedding extension metadata, we often find that data models are mismatched or that property sets overlap. In these cases, it is necessary to decide how to overcome the conflicts. When users make different decisions about how to do this, the interoperability of their metadata suffers. User communities or the bodies that create the metadata specifications can correct for this by specifying best practice guidelines

for combining the different metadata specifications. Interoperability can also be improved when users document in metadata profiles how their institution has used a metadata standard for a specific application, including which property sets and extension schemas have been used for the corresponding items in their data model. If users share their profiles by registering them with a standards editorial board, they may be reused by other potential users with similar content streams, data models, and business use cases. DePICT is a high-level model that has strived to be as much aligned with existing models as possible, so that there are as few conflicts as possible. It has also avoided defining any technology, content or organisation specific features. And it has tried to avoid arbitrary semantic distinctions while ensuring complete capture of the entities relevant for supporting digital preservation functions. It should be easily extensible and align well with potential extension schemas.

### *6.3.3 Registries*

Action has to be based on knowledge. In the digital preservation domain, this knowledge is often very technical, and expensive to create. For this reason, it is beneficial for the information to be gathered in registries so that it can be shared by all stakeholders, and so that it is possible to interoperate. In digital preservation, the main registries being created to date are

- registries of file formats, their properties and how to identify them;
- technical registries of software and hardware;
- registries of legal, statutory and policy regulations.

Registries differ substantially from repositories. Repositories record information about *actual* objects, constraints, events and agents. Registries describe *typical, abstract* objects, constraints, events and agents. Further work is needed to bring out the distinction between repositories and registries explicitly and to identify what touching points exist. For example, how do you relate specific computing platforms to abstract environments described in registries? What properties can and cannot be inherited? How do you relate concrete objects in repositories to concrete or abstract environment descriptions? There also is a continuum of generalisation between increasingly detailed environment specifications. They need to be captured through meaningful relationships, which have to be defined. This space of the types of relationships between abstract environments deserves further investigation, as it is much more diverse than the simple structural and derivative relationships that are in use between entities in repositories. Lessons may be taken from UML (Miles, Hamilton, 2006), enterprise architecture modelling approaches, such as Archimate (Open Group, 2012) and operating system specifications, such as Debian (nd).

Collecting vast amounts of data that is often hard to obtain, is necessary for creating registries that are relatively complete with respect to their users' needs. This is a daunting, and very expensive task. Collaboration between research and industry in this area is essential.

### *6.3.4 Environments as objects*

Preservation objects are part of the greater computing environment on which they depend so that they can be rendered or executed. Not all parts of this complete rendering stack are equally important for preservation. Some parts of it are subject to preservation, i.e. they must be preserved so that they remain usable in all their significant aspect or so that others remain usable through them. For some parts it is sufficient to collect metadata about them, which provides the necessary information to ensure the continued usability of the actual objects of preservation. And some parts are coincidental to preservation. They are a necessary part of the whole, but can be replaced by other components. One of the strengths of the model is that it has elevated environments to the same level of importance as preservation objects; and, in fact, they take almost identical positions and relationships in the model. The resulting question to be investigated is when one should use an `Environment` entity and when to use a `PreservationObject` entity to model parts of a computing environment that is represented as digital bit sequences. When does one choose to implement, for example, a software package as a preservation object or an environment? The answer may lie in the role it takes: If its role is to be preserved then it becomes a preservation object; if its role is a description that supports other preservation objects and environments then it will take the role of an environment. Does this however justify the creation of separate entities? Should they not be combined into one entity and just be identified through "Role" properties? Further investigation is needed how the different roles impact use of objects and environments in the preservation service cycle described in section 4. Similar considerations exist for the relationship of agents and environments when a software environment has the role of an agent in a preservation action.

# 7 APPENDICES

## 7.1 Conceptual detail

This section defines the properties of the basic entities in the conceptual model. It builds on the preceding analysis. For each concept, it describes its key attributes, and basic information such as its data type and whether it is mandatory or repeatable. It also introduces supplementary entities such as *ValueOrigin* and *Unit* that are needed to represent *Properties*.

### 7.1.1 Basic concepts

The following entities and properties are basic parts of the model.

- *Event*, *Agent*, and *Right* are entities that are adapted from PREMIS (PREMIS 2012), where *Events* and *Rights* describe *PreservationObjects* and *Agents* relate to either *Events* or *Rights*.
  A *PreservationAction* is a special kind of *Event*. A *PreservationService* is a special kind of *Agent*.
- *Description* (optional, repeatable): a human-readable, meaningful description of an entity
  - *descriptionDefinition* (optional, repeatable): A verbal definition of the entity (data constraint: string)
  - *descriptionJustification* (optional, repeatable): Why this entity is needed for digital preservation (data constraint: string)
  - *descriptionExample* (optional, repeatable): Examples (data constraint: string)
  - *descriptionNote* (optional, repeatable): Notes (data constraint: string)
  - *descriptionUsage* (optional, repeatable): How this entity is to be used (data constraint: string)

- Implementations should optimise their solutions. For example, if an organisation decides not to implement *Representations*, but rather just *IntellectualEntities* and *Files*, then the model can be adjusted to omit implicit entities and properties. The specification “mandatory” in the model refers to the necessary existence of a *property*, but not to its explicit implementation.

- Solutions to common metadata problems, such as how to address uncertain dates, open date ranges etc., are not specified in order to focus on the essential issue.

- Version information which is used to manage the history of entity instances is not included in this model. It is assumed that the system which implements this model will manage versions according to its own needs. Version information that is part of the name of the object (such as a software version or document version) is included.

### 7.1.2 Conceptual detail for preservation objects

Elements of *PreservationObject* and its Sub-classes

- *preservationObjectIdentifier* (mandatory, non-repeatable): a unique identifier of the *PreservationObject* (data constraint: *PreservationObject* ID)
- *preservationObjectName* (optional, repeatable): a human-readable, meaningful descriptor for the *PreservationObject* (data constraint: string)
- *preservationObjectDescription* (optional, repeatable): a human-readable, meaningful description for the *PreservationObject* (data constraint: Description)
- *hasRelatedObject* (optional, repeatable): a unique identifier of a related object (data constraint: *PreservationObject* ID)
- *hasEnvironment* (optional, repeatable): unique identifiers of each of the *PreservationObject*'s *Environments* (data constraint: *Environment* ID)
- *hasCharacteristic* (optional, repeatable): unique identifiers of each of the *Characteristics* of the *PreservationObject* (data constraint: *Characteristic* ID). Every *PreservationObject* has none or more *Characteristics* with associated *Values* which may influence the choice of *PreservationAction*.
- *hasRisk* (optional, repeatable): unique identifiers of each of the *PreservationRisks* which have arisen as the *PreservationObject*'s *Characteristics* violate a *RiskSpecifyingConstraint* (data constraint: *PreservationRisk* ID).
- *hasPolicy* (optional, repeatable): unique identifiers of each of the *PreservationObject*'s *Policies* (data constraint: *Policy* ID)
- *hasRight* (optional, repeatable): unique identifiers of each of the *PreservationObject*'s *Rights* objects (data constraint: *Rights* ID)
- *hasStakeholder* (optional, repeatable): unique identifiers of each of the *PreservationObject*'s stakeholders (data constraint: *Agent* ID)

- *isInputPreservationObjectTo* (optional, repeatable): unique identifiers of each of the *PreservationObject*'s *PreservationAction* objects to which it is an input object (data constraint: *PreservationAction* ID)

- *isOutputPreservationObjectTo* (optional, repeatable): unique identifiers of each of the *PreservationObject*'s *PreservationAction* objects to which it is an output object (data constraint: *PreservationAction* ID)

- *hasEvent* (optional, repeatable): unique identifiers of each of the *PreservationObject*'s *Event* objects, other than *PreservationAction Events* (data constraint: *Event* ID) [33]

Other relationships of *PreservationObject*

- *TransformationPreservationAction* has a *hasInputPreservationObject* and a *hasOutputPreservationObject* relationship with *PreservationObject*.

- *Characteristic* and *PreservationRisk* have an *associatedWith* relationship with *PreservationObject*.

- *Policy* has a *hasPreservationObject* relationship with *PreservationObject*.

Vocabulary for *PreservationObject* Sub-classes

- Extensible vocabulary including *IntellectualEntity*, *Representation* and *Bitstream*
- They can be further categorised as illustrated earlier in section 3.1.1.

#### *7.1.2.1 Conceptual detail for intellectual entities*

[33] For all events the following holds: Whether recording a certain event is mandatory, and which event to record is a business requirement of the institution. It is not made mandatory by the data model.

Elements of IntellectualEntity and its Sub-classes

- Elements inherited from *PreservationObject*.
- *hasRepresentation* (optional, repeatable): unique identifier of the *IntellectualEntity*'s *Representation*.

Other Relationships with IntellectualEntity

- *IntellectualEntity* is a sub-class of *PreservationObject*.
- *Representation* has a *embodiesIntellectualEntity* relationship with *IntellectualEntity*.

### *7.1.2.2 Conceptual detail for representations*

Elements of Representation

- Elements inherited from *PreservationObject*
- *embodiesIntellectualEntity* (mandatory, repeatable): unique identifier of the *IntellectualEntity* for which the *Representation* serves as physical embodiment. (data constraint: *IntellectualEntity* ID)
- *hasRepresentationBitstream* (mandatory, repeatable): unique identifier of the *Bitstreams* that make up the *Representation*. (data constraint: *RepresentationBitstream* ID)
- *hasRepresentationBitstreamStructmap* (mandatory, repeatable): information to capture the physical and logical structural relationships of the *RepresentationBitstreams* that make up the *Representation*. See the METS structMap definition for comparison. (METS, nd).

Other Relationships with Representation

- *Representation* is a sub-class of *PreservationObject*.
- *IntellectualEntity* has a *hasRepresentation* relationship with *Representation*.

- *RepresentationBitstream* has a *hasRepresentation* relationship with *Representation*

#### *7.1.2.3 Conceptual detail for representation bitstreams*

Elements of RepresentationBitstream

- Elements inherited from *PreservationObject.*
- *hasRepresentation* (mandatory, repeatable): unique identifier of each of the *RepresentationBitstream*'s *Representations* (data constraint: *Representation* ID).
- *implementedBy* (mandatory, non-repeatable): unique identifier of the physical object that implements the *Bitstream* including offset information, etc. (data constraint: *Bitstream* ID).

Other Relationships with RepresentationBitstream

- *Representation* has a *hasRepresentationBitstream* relationship with *RepresentationBitstream*.
- *Bitstream* has a *implements* relationship with *RepresentationBitstream*.

#### *7.1.2.4 Conceptual detail for bitstreams*

Elements of Bitstream

- Elements inherited from *PreservationObject*
- *implements* (mandatory, repeatable): unique identifier of the *RepresentationBitstreams* which are realised by the *Bitstream* (data constraint: *RepresentationBitstream* ID).

Other Relationships with Bitstream

- *Bitstream* is a sub-class of *PreservationObject*.

- *RepresentationBitstream* has a *implementedBy* relationship with Bitstream.

### 7.1.3 Conceptual detail for environments

Elements of Environment and its Sub-classes

*Environment* can take on the role of a *PreservationObject* and, in that case, has general *PreservationObject* properties.

- *environmentIdentifier* (mandatory, non-repeatable): a unique identifier of the *Environment* (data constraint: *Environment* ID)
- *environmentName* (optional, repeatable): a human-readable, meaningful descriptor for the *Environment* (data constraint: string)
- *environmentDescription* (optional, repeatable): a human-readable, meaningful description for the *Environment* (data constraint: *Description*)
- *environmentPurpose* (optional, repeatable): (data constraint: extensible vocabulary: taken from *creation*, *ingest*, *preservation*, *remote access*, *local access*, *migration*,…)
- *environmentFunction* (optional, repeatable): (data constraint: extensible vocabulary: taken from *rendering, editing, executing, printing….)*
- *environmentIntention* (optional, repeatable): (data constraint: extensible vocabulary: taken from *necessary*, *recommended*, *acceptable*…)
- *hasRelatedEnvironment* (optional, repeatable): unique identifiers to each related *Environment* objects (data constraint: *Environment* ID)
- *isEnvironmentOf* (optional, repeatable): a unique identifier of the *PreservationObjects* to which the *Environment* belongs; the type of relationship between the two Entities needs to be captured, e.g. is the *Environment* necessary for rendering the *PreservationObject*, is the *PreservationObject* an implementation of the *Environment, etc.?* (data constraint: *PreservationObject* ID)
  (Inverse of the *hasEnvironment* relationship from *PreservationObject* to *Environment*).

- *isInputEnvironmentTo* (optional, repeatable): unique identifiers of each of the *TransformationPreservationActions* to which it is an input *Environment* (data constraint: *TransformationPreservationActions* ID)

- *isOutputEnvironmentTo* (optional, repeatable): unique identifiers of each of the *TransformationPreservationActions* to which it is an output *Environment* (data constraint: *TransformationPreservationActions* ID)

- *isEnvironmentOf* (optional, repeatable): a unique identifier of the *PreservationActions* to which the *Environment* belongs; (data constraint: *PreservationAction* ID).
  (Inverse of the *hasEnvironment* relationship from *PreservationObject* to *Environment*).

- *actsAsPreservationService* (optional, repeatable): a unique identifier of the *PreservationService* which is provided by the *Environment* in its role as a *Tool*; (data constraint: *PreservationService* ID).
  (Inverse of the *isPreservationServiceAgent* relationship from *PreservationService* to *Environment*.
  The same semantics are captured by the combination of the *isEnvironmentOf* relationship between the *Environment* and a *PreservationAction* and the *hasPreservationService* relationship between this *PreservationAction* and its *PreservationService*.).

- *hasCharacteristic* (optional, repeatable): unique identifier of each of the *Characteristics* of the *Environment* (data constraint: *Characteristic* ID). Every *Environment* has none or more *Characteristics* with associated *Values* which may influence the choice of *PreservationAction*.

- *hasRisk* (optional, repeatable): unique identifiers of each of the *PreservationRisks* which have arisen as the *Environment's Characteristics* violate a *RiskSpecifyingConstraint* (data constraint: *PreservationRisk* ID).

- *hasPolicy* (optional, repeatable): unique identifiers of each of the *Environment*'s *Policies* (data constraint: *Policy* ID).

- *hasRight* (optional, repeatable): unique identifiers of each of the *Environment*'s *Rights* objects (data constraint: *Rights* ID).

- *hasEvent* (optional, repeatable): unique identifiers to each of the *Environment*'s *Event* objects (data constraint: *Event* ID).

Other Relationships with Environment

- *PreservationObject* has a *hasEnvironment* relationship with *Environment*.
- *PreservationAction* has a *hasEnvironment*, a *hasInputEnvironment* and a *hasOutputEnvironment* relationship with *Environment*.
- *Policy* has a *hasEnvironment* relationship with *Environment*.
- *Rights* has a *isRightOf* relationship with *Environment*.
- *Event* has a *hasEnvironment* relationship with *Environment*.
- *Characteristic* and *PreservationRisk* have an *associatedWith* relationship with *Environment*.

### 7.1.4 Conceptual detail for properties

Elements of *Property*

- *propertyIdentifier* (mandatory, non-repeatable): a unique identifier of the *Property* (data constraint: *Property* ID).
- *propertyName* (optional, repeatable): a meaningful human-readable name for the *Property* (data constraint: string). It is repeatable in order to allow for synonyms. Different *Properties* may have the same names, but must have unique identifiers.
- *propertyDescription* (optional, repeatable): a human-readable, meaningful description for the *Property* (data constraint: *Description*)
- *appliesTo* (mandatory, non-repeatable): domain specification;
  vector of *PreservationObject*, *Environment* or *PreservationAction* sub-classes to which the *Property* applies. It can be meaningfully associated with instances of these classes;
  "n-ary parameter list" (data constraint: vector of *PreservationObject*, *Environment* or *PreservationAction* sub-classes).
  The vocabulary of sub-classes can be extensible and include many sub-classes not shown in this work. Some vocabulary for sub-classes can be found in Figure 23, Figure 24, Figure 25, Figure 26, Figure 27 and Figure 36.
- *hasRange* (optional, repeatable): range specification; *Constraints* on or enumeration of permissible *Values*; a data type definition for the *Value*; possibly a URI pointing to the defined vocabulary for the *Property*
  - *hasUnit* (optional, non-repeatable): See section 3.1.4.3. A unique identifier of the *Unit*
  - *hasDataConstraint* (mandatory, non-repeatable): permissible *Values*; a type definition for the *Value*; possibly a URI for defined vocabulary for the *Property* (data constraint: taken from an extensible set of data constraints). Data constraints are combined with the *Unit* definition, as different *Units* may have different data constraints. (E.g. K: ≥0, °C: ≥ -273.15, °F: ≥ -459.67). It has to be compatible with the *Unit*'s data constraint

- *isDefault* (optional, non-repeatable): indicates whether this range specification is the default range for this *Property* (data constraint: “yes” or “no”)
- *hasDefaultValue* (optional, non-repeatable): a default *Value* for this *Property*

- *hasValueOrigin* (optional, repeatable): How the *Value* for the *Property* may be obtained or updated (if it is stored)
  - *hasValueOriginID* (mandatory, non-repeatable): a unique identifier of the *ValueOrigins*. See section 3.1.4.2
  - *isDefault* (optional, non-repeatable): indicates whether this *ValueOrigin* is the default for this *Property* (data constraint: “yes” or “no”)

- *hasRelationship* (optional, repeatable): relationship to other *Property* classes with related semantics (relationships that are not captured by has*ValueOrigin*)
  - *hasRelatedProperty* (mandatory, non-repeatable): (data constraint: *Property* ID)
  - *hasRelationshipType* (mandatory, non-repeatable): a type specification of the relationship to another *Property* class (data constraint: local usage, such as *generalisationOf*, *specialisationOf*, *siblingOf*, *inverseOf*, *disjointOf*, *smallerThan*, or any association name)

Other Relationships with *Property*

- *Characteristic* has a *hasProperty* relationship with *Property*.

### 7.1.5 Conceptual detail for value origins

The *ValueOrigin* entity provides a way to specify where a particular *Value* comes from or how it can be obtained. There are multiple ways of obtaining the *Value* of a *Property* that do not produce conflicting results. For example, they might be measured from different sources, measured by different techniques, measured using different tools, or obtained through different agents.

Elements of ValueOrigin

- *valueOriginIdentifier* (mandatory, non-repeatable): a unique identifier of the *ValueOrigin* (data constraint: none)
- *valueOriginName* (optional, repeatable): a human-readable, meaningful descriptor for the *valueOrigin* (data constraint: string)
- *valueOriginDescription* (optional, repeatable): a human-readable, meaningful description for the *ValueOrigin*. (data constraint: *Description*)
- *hasSource* (optional, repeatable): a type specification of the sources from which the *Value* can be measured or derived (data constraint: none). Sources for the *Value* may be registries or inventories, *Values* for other *Properties* from which the *Value* can be derived (In that case the source would have to be a list of parameter definitions including the *Unit* and *ValueOrigin* of the source-*Properties*), or *Representations* of the *IntellectualEntities* and their *Environments* from which the *Value* can be measured. There may be a chain of *ValueOrigins* where one *ValueOrigin* is the source for another.
- *hasTargetUnit* (optional, repeatable): a specification of the Unit of the *Value* to be created by this *ValueOrigin*. (data constraint: *Unit* ID)
- *hasTechnique* (optional, repeatable): Rule, algorithm or logic used for obtaining the *Value* (e.g. assigned according to Anglo-American Cataloguing Rules, extracted from .tiff *File* metadata) (data constraint: none). Dependent on whether the *Value* is created manually or automatically different preservation processes need to be used.
  One special *technique* is the specification of a conversion. Conversions specify how a *Value* for the *Property* may be derived from other

*Properties* for specified *Units* and *ValueOrigins*; or from a related *Property* obtained by other *ValueOrigins*, since the related *Properties* can have slightly different measurement results when measured using different *ValueOrigins*, e.g. through systematic errors.

- *hasAgent* (optional, repeatable): For automatically derived *Values*: software service and version; for manually assigned *Values*: person or role (data constraint: *Agent* ID). There may be multiple possible *Agents*.
- *hasTrigger* (optional, repeatable): a trigger for *Value* assignment: e.g. ingest, *PreservationService*, etc. (data constraint: none)

Other Relationships with ValueOrigin

- *Property* has a *hasValueOrigin* relationship *ValueOrigin*.

### 7.1.6 *Conceptual detail for units*

Elements of Unit

- *unitIdentifier* (mandatory, non-repeatable): a unique identifier of the *Unit* (data constraint: none)
- *unitName* (optional, repeatable): (data constraint: string) allows for synonyms; e.g. inches, Zoll
- *unitDescription* (optional, repeatable): a human-readable, meaningful description for the *Unit* (data constraint: *Description*)
- *hasDataConstraint* (mandatory, non-repeatable): permissible *Values*; a type definition for the *Value*; possibly a URI for defined vocabulary (data constraint: taken from an extensible set of data constraints)
- *hasConversion* (optional, repeatable): How *Values* may be converted from another *Unit* to this *Unit* . This is important for preservation characterisation and comparison.
    - *hasSource* (mandatory, non-repeatable): Identifier of the source *Unit* (data constraint: *Unit* ID)
    - *hasTechnique* (mandatory, repeatable): Rule, algorithm or logic used for mapping or converting the *Value* (e.g. FFT) (data constraint: none). There may be multiple ways of deriving the *Value*.
    - *hasAgent* (optional, repeatable): conversion software tool and version; (data constraint: *Agent* ID). There may be multiple possible *Agents*.

Other Relationships with Unit

- *Property*, *Characteristic* and *Constraint* have a *hasUnit* relationship with *Unit*.

### 7.1.7 *Conceptual detail for characteristics*

Elements of Characteristic

- *characteristicIdentifier* (mandatory, non-repeatable): a unique identifier of the *Characteristic*. Having a unique identifier for a *Characteristic* supports having different *Values* for the same *Property* at different times. (data constraint: *Characteristic* ID)
- *associatedWith* (mandatory, non-repeatable):
  vector of unique identifiers of *PreservationObject*, *Environment* or *PreservationAction* instances with which the *Characteristic* is associated. It can be meaningfully associated with instances of the classes defined in the *appliesTo* property of the corresponding *Property* class (data constraint: vector of *PreservationObject*, *Environment* or *PreservationAction* IDs).
  (This relationship is also established via the *hasCharacteristic* relationship of *PreservationObject*, *Environment*, or *PreservationAction*).
- *hasProperty* (mandatory, non-repeatable): a specification of the *Property* to which this *Characteristic* refers
  - *PropertyIdentifier* (mandatory, non-repeatable): It specifies for which *Property* the *Characteristic's Value* holds (data constraint: *Property* ID).
  - *annotations* (optional, non-repeatable): chosen from the allowable *Values* specified in the corresponding *Property* definition.
- *hasUnit* (optional, non-repeatable): a unique identifier of the *Unit* of the *Value* (data constraint: *Unit* ID).
- *hasValueOrigin* (optional, non-repeatable): a unique identifier of the *ValueOrigin* which specifies how the *Value* is to be obtained on demand or was obtained, if stored. (data constraint: *ValueOrigin* ID). The *technique, source* and *agent* employed must be of the types specified for the *ValueOrigin* and *Property*.
    - *hasSource* (optional, non-repeatable)

- *hasTechnique* (optional, non-repeatable)
- *hasAgent* (optional, non-repeatable)

- *onDemand* (optional, non-repeatable): a specification of whether the *Value* is stored locally or should be derived on demand (data constraint: taken from *local, onDemand)*. Registry look-up can be considered an on-demand access.
- *hasValue* (optional, non-repeatable): *Value* of the *Characteristic*, if it is stored locally (data constraint: none).
- *hasCreationEvent* (optional, non-repeatable): a unique identifier of the *Event* which created the *Value* if it is stored locally (data constraint: *Event* ID) including the date the *Value* was set. In addition, information to capture versioning information such as a date range of applicability of the *Value*, previous *Values* for the same *Property* and objects, etc. will be desirable.

Other Relationships with Characteristic

- *PreservationObject*, *Environment*, and *PreservationAction* have a *hasCharacteristic* relationship with *Characteristic*.

### 7.1.8 Conceptual detail for preservation risks

Elements of PreservationRisk

- *riskIdentifier* (mandatory, non-repeatable): a unique identifier of the *PreservationRisk* (data constraint: *PreservationRisk* ID).
- *riskName* (optional, repeatable): a human-readable, meaningful descriptor for the *PreservationRisk* (data constraint: string).
- *riskDescription* (optional, repeatable): a human-readable, meaningful description for the *PreservationRisk* (data constraint: *Description*).
- *associatedWith* (mandatory, non-repeatable):
  vector of unique identifiers of *PreservationObject* or *Environment* instances that are at risk
  (data constraint: vector of *PreservationObject* or *Environment* IDs). A *PreservationRisk* may consist of the interplay of 2 or more *PreservationObjects* or *Environments*. Therefore there is a need to express a vector of affected Entity instances. It can be meaningfully associated with instances of the Classes defined in the *hasConstraint* element of *PreservationRisk*.
  (Inverse of the *hasRisk* relationship from *PreservationObject* or *Environment* to *PreservationRisk*).
- *hasConstraint* (mandatory, non-repeatable): a unique identifier of the *RiskSpecifyingConstraint* which is violated by the *PreservationRisk* (data constraint: *RiskSpecifyingConstraint* ID).
- *hasEvent* (optional, repeatable): unique identifiers to each of the *PreservationRisk's Event* instances (data constraint: *Event* ID). This might have specific information about which *Characteristics* of which *PreservationObject* or *Environment* violated the *RiskSpecifyingConstraint* at the time when the *PreservationRisk* was discovered.

Other Relationships with PreservationRisk

- *Environment* and *PreservationObject* have a *hasRisk* relationship with *PreservationRisk*.

### 7.1.9 Conceptual detail for preservation actions

Elements of PreservationAction

*PreservationAction* is an *Event* and inherits general *Event* Properties (see PREMIS), such as start time / end time, agent, and outcome. It has additional elements.

- *actionIdentifier* (mandatory, non-repeatable): a unique identifier of the concrete *PreservationAction* (data constraint: *PreservationAction* ID).
- *actionName* (optional, repeatable): a human-readable, meaningful descriptor for the *PreservationAction* (data constraint: string).
- *actionDescription* (optional, repeatable): a human-readable, meaningful description for the *PreservationAction*. This is not a description of a *PreservationTool* or *PreservationService*, but a description of the actual *PreservationAction Event*. (data constraint: *Description*).
- *relatedTo* (optional, repeatable): unique identifiers of each of the related *PreservationAction* objects for composite *PreservationActions* (data constraint: *PreservationAction* ID). Expressing relationships between composite *PreservationActions* adequately requires the expressiveness of business process modelling languages.
- *hasRisk* (optional, repeatable): a unique identifier of the concrete *PreservationRisk* which prompts the *PreservationAction* (data constraint: *PreservationRisk* ID). The *PreservationRisk* instance contains the information about the violated *RiskSpecifyingConstraint* and the *Environments* or *PreservationObjects* that are at risk.
- *hasInputPreservationObject* (optional, non-repeatable): a unique identifier of the *PreservationObject* on which the *PreservationAction* is being executed (data constraint: *PreservationObject* ID). It is optional since a *PreservationAction* might only address an *Environment*.

- *hasOutputPreservationObject* (optional, non-repeatable): a unique identifier of the output *PreservationObject* which results from the execution of a *TransformationPreservationAction* (data constraint: *PreservationObject* ID).
- *hasInputEnvironment* (optional, non-repeatable): a unique identifier of the applicable *Environment* of the input *PreservationObject* (data constraint: *Environment* ID) including all sub-*Environments* and their *Characteristics* which can be used to evaluate *PreservationGuidingConstraints*.
- *hasOutputEnvironment* (optional, non-repeatable): a unique identifier of the *Environment* of the output *PreservationObject* (data constraint: *Environment* ID) including all sub-*Environments* and their *Characteristics* which the *PreservationObject* would have after execution of a *TransformationPreservationAction*. These can be used to evaluate *PreservationGuidingConstraints*.

At least one input and output *PreservationObject* or *Environment* has to exist for the *TransformationPreservationAction* to affect a state change.

- *hasEnvironment* (optional, repeatable): a unique identifier of each of the *Environments* of the *PreservationAction* itself (data constraint: *Environment* ID). They can be used to evaluate *ActionDefiningConstraints*.
- *hasPreservationService* (optional, repeatable): a unique identifier of the *PreservationService* which executes the *PreservationAction*; (data constraint: *PreservationAction* ID).
- *hasCharacteristic* (optional, repeatable): unique identifier of each of the *Characteristics* of the *PreservationAction* (data constraint: *Characteristic* ID). Every *PreservationAction* has none or more *Characteristics* with associated *Values*.
- *hasEventOutcome* (optional, repeatable): in addition to PREMIS *Event* outcomes:
    - *hasPolicy* (optional, repeatable): unique identifier of the sets of *Constraints* under which this *PreservationAction* has been performed.

- *degreeOfCompliance* (optional, repeatable): specifies for a *Constraint* to what degree and by what measure the *PreservationAction* complied with the *Constraint*. A *PreservationAction* can store to what degree the *Constraints* have been satisfied.
  - *hasConstraint* (mandatory, non-repeatable): (data constraint: *Constraint* ID).
  - *associatedWith* (mandatory, non-repeatable):
    vector of unique identifiers of affected *PreservationObject* and/or *Environment* instances (data constraint: vector of *PreservationObject* or *Environment* IDs).
  - *hasMeasure* (): (data constraint: none).
  - *hasOutcome* (): (data constraint: none).

Other Relationships with PreservationAction

- *PreservationObject* has an *isInputPreservationObjectTo* relationship with *TransformationPreservationAction*.
- *PreservationObject* has an *isOutputPreservationObjectTo* relationship with *TransformationPreservationAction*.
- *Environment* has an *isInputEnvironmentTo* relationship with *TransformationPreservationAction*.
- *Environment* has an *isOutputEnvironmentTo* relationship with *TransformationPreservationAction*.
- *Environment* has an *isEnvironmentOf* relationship with *PreservationAction*.
- *PreservationService* has an *isPreservationServiceOf* relationship with *PreservationAction*.
- *Characteristic* has an *associatedWith* relationship with *PreservationAction*.

### 7.1.10 Conceptual detail for policies

| Elements of Policy |
|---|

- *policyIdentifier* (mandatory, non-repeatable): a unique identifier of the *Policy* (data constraint: *Policy* ID).
- *policyName* (optional, repeatable): a human-readable, meaningful descriptor for the *Policy* (data constraint: string).
- *policyVersion* (optional, non-repeatable): Version of the *Policy* (data constraint: none).
- *StratML:Organization* (mandatory, non-repeatable): a unique identifier of the organisation (data constraint: *Agent* ID).
- *policyApproval* (optional, repeatable).
    - *status* (mandatory, non-repeatable): (data constraint: taken from *proposed*, *approved*, *superseded*, *withdrawn*).
    - *initiator* (optional, repeatable): Person who proposed, approved or withdrew the *Policy*. (data constraint: *Agent* ID) (N.B. This subsumes the *StratML:submitter* element).
    - *statusDate* (mandatory, non-repeatable): Date on which the *Policy* was proposed, approved or withdrawn. (N.B. This subsumes the *StratML:Date* attribute).
- *policyApplicability* (mandatory, non-repeatable).
    - *startDate* (optional, non-repeatable): The date the *Policy* is projected to become valid (data constraint: date).
    - *endDate* (optional, non-repeatable): The date the *Policy* is projected to cease, if it is not subsequently extended (data constraint: date).
- *StratML:Source* (optional, non-repeatable): The Web address (URL) for the authoritative source of this *Policy* (data constraint: anyURI).

- *StratML:Vision* (optional, repeatable): Vision statements are distinguished from goals in that they are the focus of constant pursuit but can never be satisfied in the sense of being met or completed. A concise and inspirational description of a state the organisation will strive to approach over a relatively long span of years but which can ultimately never be fully achieved. (data constraint: string).
- *StratML:Mission* (optional, repeatable): Mission Statement. A brief description of the basic purpose of the organisation. An agency's goals should flow from the mission statement. (data constraint: string).
- *StratML:Value* (optional, repeatable): A principle that is important and helps to define the essential character of the organisation.
    - *StratML:Name* (optional).
    - *StratML:Description* (optional, repeatable).
- *Goal* (mandatory, repeatable): General Goal. A relatively broad statement of intended results to be achieved over more than one resource allocation and performance measurement cycle. Goals define a purpose and direction and take all stakeholders and perceived present and future needs into account. Goals must be capable of being effectively pursued with measurable results over more than one budgetary execution cycle but within the reasonably foreseeable future. Goals should be objective, quantifiable, measureable, and defined at the level to be achieved by a program activity. Supports Mission.
    - *SequenceIndicator* (optional, non-repeatable).
    - *StratML:Name* (optional, non-repeatable).
    - *StratML:Description* (mandatory, non-repeatable) (data constraint: *Description*).
    - *StratML:Stakeholder* (optional, repeatable) (data constraint: *Agent* ID).
    - *hasConstraint* (optional, repeatable): a unique identifier of the *Constraints* included in this *policy* (data constraint: *Constraint* ID).

- *StratML:OtherInformation* (optional, non-repeatable).

- *references* (optional, repeatable).
  - *hasPreservationObject* (optional, repeatable): unique identifiers for each of the organisation's *PreservationObjects* (which can be a set of *PreservationObjects*, as in a collection) to which the *Policy* refers (data constraint: *PreservationObject* ID).
  - *hasEnvironment* (optional, repeatable): unique identifiers for each of the organisation's *Environments* to which the *Policy* refers (data constraint: *Environment* ID).
  - *hasRegistryReference* (optional, repeatable): unique identifiers for each of the registries and inventories to which the *Policy* refers (data constraint: *Registry* ID).
  - *hasPredecessorPolicy* (optional, repeatable): unique identifiers for each of the predecessor *Policy*(s) of the *Policy* (data constraint: *Policy* ID).
  - *hasRelatedPolicy* (optional, repeatable): unique identifiers for each of other related *Policy* (data constraint: *Policy* ID).

Other Relationships with Policy

- *PreservationObject* and *Environment* and *Constraint* have a *hasPolicy* relationship with *Policy*.

### *7.1.11 Conceptual detail for constraints*

| Elements of Constraints |
|---|

- *constraintIdentifier* (mandatory, non-repeatable): a unique identifier of the *Constraint* (data constraint: *Constraint* ID).
- *constraintName* (optional, repeatable): a human-readable, meaningful name for the *Constraint* (data constraint: string).
- *constraintDescription* (optional, repeatable): a human-readable, meaningful description for the *Constraint* (data constraint: Description).
- *hasPolicy* (optional, repeatable): a unique identifier of the *policy* to which the *Constraint* belongs (data constraint: *Policy* ID).
- *hasStakeholder* (optional, repeatable): (data constraint: *Agent* ID).
- *constraintSource* (optional, repeatable).
- *constraintApplicability* (optional, non-repeatable): Time range during which the *Constraint* is applicable. If it is not specified explicitly, then it defaults to the *Value* of the *applicability* element of the *Policy* in which the *Constraint* is captured.
    - *startDate* (optional, non-repeatable): The date the *Constraint* is projected to become valid (data constraint: date).
    - *endDate* (optional, non-repeatable): The date the *Constraint* is projected to cease, if it is not subsequently extended (data constraint: date).
- *constraintSpecification* (mandatory, non-repeatable):
    - *context* (optional, repeatable): Specifies the objects for which the constraint holds.
    - *pre* (optional, non-repeatable): Specifies a pre-condition for applying the constraint.
    - *post* (optional, non-repeatable): Specifies a post-condition for applying the constraint.

The *constraintSpecification* element can be modelled similar to constraint languages such as OCL (OCL 2003). Each pre- and post-condition is a logical expression which combines constraints and can be evaluated to true or false for a given set of *Characteristics Values*.

In general a constraint will contain some of the following parts:

- *operator*: Operator to be applied to determine whether the *Constraint* is satisfied.
  - *operator* (mandatory, non-repeatable): Function to be evaluated. e.g. "=", "one of", "MyBooleanFunction". The function should evaluate to true/false. If a tolerance is specified the function might return the degree to which the *Constraint* is satisfied with respect to the tolerance.
  - *tolerance* (optional, non-repeatable): To what degree deviation from the *Constraint* can be tolerated.
- *Property*: It specifies for which *Property* a *Value* should be retrieved. A *Property* is fully specified by the following elements.
  - *PropertyIdentifier* (mandatory, non-repeatable): It specifies for which *Property* a *Value* should be retrieved.
  - *annotations* (optional, non-repeatable): It specifies which of the annotations listed within the *Property* definition should be used to derive the *Value*.
    - *hasUnit* (optional, non-repeatable): a unique identifier of the *Unit* of the *Value* (data constraint: *Unit* ID)
    - *hasValueOrigin* (optional, non-repeatable): a unique identifier of the *ValueOrigin* which specifies how the *Value* is to be or was obtained. (data constraint: *ValueOrigin* ID).

      The *technique*, *source* and *agent* employed must be of the types specified for the *ValueOrigin* and *Property*.
    - *hasSource* (optional, non-repeatable)
    - *hasTechnique* (optional, non-repeatable)

        - *hasAgent* (optional, non-repeatable)

    - *constant*: It specifies a constant *Value*. A constant is fully specified by the following two elements.

        - *value* (mandatory, non-repeatable).
        - *unit* (mandatory if applicable, non-repeatable).

        *Units* in *constraintsSpecifications* have to agree with *Units* of *Characteristics Values* and the *Property*'s *hasUnit* (must be the same or have a conversion specified).

*constraintImportanceFactor:* Measure of the relative significance of the *Constraint* for the stakeholder (data constraint: none).

- *hasEvent* (optional, repeatable): unique identifiers to each of the *Constraint's Event* objects (data constraint: *Event* ID).

The *constraintImportanceFactor* and the *tolerance* elements allow for computing a weighted measure of compliance with the *Constraint*.

The *constraintSpecification* can be expressed informally or implemented using a constraint language such as OCL (Warner, Kleppe, 2003). In the latter case, each pre and post-condition is an expression that can be evaluated against the *Characteristic Values* specified in the *Constraint's* context. In some implementations, these will evaluate to simple Boolean *Values* (true or false). Other implementations will allow for a *tolerance*. In this case, the *constraintImportanceFactor* and *tolerance* can be used to compute a weighted measure of compliance with the *Constraint*.

Other Relationships with Constraint

- *Policy* has a *hasConstraint* relationship to *Constraint*.
- *PreservationRisk* has a *hasRiskSpecifyingConstraint* association relationship to *RiskSpecifyingConstraint*.
- *PreservationAction* has a *hasConstraint* relationship to *Constraint* in its *hasEventOutcome* Property.

## 7.2 Machine-interpretable model

This section contains an xsd implementation of the data dictionary.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.planets-project.eu/pp2" elementFormDefault="qualified"
    attributeFormDefault="unqualified" xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:stratml="http://www.stratml.net" xmlns:pp2="http://www.planets-project.eu/pp2"
    xmlns:premis="http://www.loc.gov/standards/premis/v1">
    <import namespace="http://www.stratml.net"
            schemaLocation="http://xml.gov/stratml/draft/StrategicPlan.xsd"/>
    <import namespace="http://www.loc.gov/standards/premis/v1"
            schemaLocation="http://www.loc.gov/standards/premis/v1/PREMIS-v1-1.xsd" />

<complexType name="PolicyType">
        <sequence>
            <element name="policyName" type="string" minOccurs="0" maxOccurs="unbounded"/>
            <element name=" policyVersion " minOccurs="0" maxOccurs="unbounded"/>
            <element name="policyApproval" maxOccurs="unbounded" minOccurs="0">
                <complexType>
                    <sequence>
                        <element name="initiator" maxOccurs="unbounded" minOccurs="0" type="IDREF"/>
                        <!-- Agent ID -->
                    </sequence>
                    <attribute name="status" type="string" use="required"/>
                    <!-- proposed, approved, superseded -->
                    <attribute name="statusDate" type="date" use="required"/>
                </complexType>
            </element>
            <element name="policyApplicability" maxOccurs="1" minOccurs="1">
                <complexType>
                    <attribute name="startDate" type="date" use="optional"/>
```

```
                <attribute name="endDate" type="date" use="optional"/>
        </complexType>
</element>
<element ref="stratml:Organization" minOccurs="0" maxOccurs="1"/>
<element ref="stratml:Source" minOccurs="0" maxOccurs="1"/>
<element ref="stratml:Vision" minOccurs="0" maxOccurs="unbounded"/>
<element ref="stratml:Mission" minOccurs="0" maxOccurs="unbounded"/>
<element ref="stratml:Value" minOccurs="0" maxOccurs="unbounded"/>
<element name="Goal" minOccurs="1" maxOccurs="unbounded">
        <complexType>
                <sequence>
                        <element ref="stratml:SequenceIndicator" minOccurs="0" maxOccurs="1"/>
                        <element ref="stratml:Name" minOccurs="0" maxOccurs="1"/>
                        <element ref="stratml:Description" minOccurs="1" maxOccurs="1"/>
                        <element ref="stratml:Stakeholder" minOccurs="0" maxOccurs="unbounded"/>
                        <element name="hasConstraint" minOccurs="0" maxOccurs="unbounded">
                                <complexType>
                                        <attribute name="idref" type="IDREF" use="required"/>
                                        <!-- Constraint ID -->
                                </complexType>
                        </element>
                        <element ref="stratml:OtherInformation" minOccurs="0" maxOccurs="1"/>
                </sequence>
        </complexType>
</element>
<element name="references" maxOccurs="unbounded" minOccurs="0">
        <complexType>
                <sequence>
                        <element name=" hasPreservationObject" minOccurs="0" maxOccurs="unbounded">
                                <complexType>
                                        <attribute name="idref" type="IDREF" use="required"/>
                                </complexType>
```

```
                                        <!-- Type should be ID of a set of PreservationObjects -->
                                    </element>
                                    <element name="hasRegistryReference" minOccurs="0" maxOccurs="unbounded">
                                        <complexType>
                                            <attribute name="idref" type="IDREF" use="required"/>
                                            <!-- Registry ID -->
                                        </complexType>
                                    </element>
                                    <element name="hasPredecessorPolicy" minOccurs="0"
                                        maxOccurs="unbounded">
                                        <complexType>
                                            <attribute name="idref" type="IDREF" use="required"/>
                                            <!—Policy ID -->
                                        </complexType>
                                    </element>
                                    <element name="hasRelatedPolicy" minOccurs="0"
                                        maxOccurs="unbounded">
                                        <complexType>
                                            <attribute name="idref" type="IDREF" use="required"/>
                                            <!-- Policy ID -->
                                        </complexType>
                                    </element>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                    <attribute name="id" type="ID" use="required"/>
                    <!-- Policy ID -->
                </complexType>

                <complexType name="PreservationObjectType">
                    <sequence>
```

```
<element name="preservationObjectName" type="string" maxOccurs="unbounded" minOccurs="0"/>
<element name="preservationObjectDescription" type="string" minOccurs="0"
        maxOccurs="unbounded"/>
<element name="hasRelatedObject" minOccurs="0" maxOccurs="unbounded">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- PreservationObjectID -->
        </complexType>
</element>
<element name="hasEnvironment" minOccurs="0" maxOccurs="unbounded">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- Environment ID -->
        </complexType>
</element>
<element name="hasCharacteristic" minOccurs="0" maxOccurs="unbounded">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- Characteristic ID -->
        </complexType>
</element>
<element name="hasRisk" minOccurs="0" maxOccurs="unbounded">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- PreservationRisk ID -->
        </complexType>
</element>
<element name="hasPolicy" minOccurs="0" maxOccurs="unbounded">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- Policy ID -->
        </complexType>
```

```
                </element>
                <element name="hasStakeholder" minOccurs="0" maxOccurs="unbounded">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- Agent ID -->
                        </complexType>
                </element>
                <element name="hasRight" minOccurs="0" maxOccurs="unbounded">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- Right ID -->
                        </complexType>
                </element>
                <element name="hasEvent" minOccurs="0" maxOccurs="unbounded">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- Event ID -->
                        </complexType>
                </element>
        </sequence>
        <attribute name="id" type="ID" use="required"/>
        <!-- PreservationObject ID -->
</complexType>

<complexType name="IntellectualEntityType">
        <complexContent>
                <extension base="pp2:PreservationObjectType">
                        <sequence>
                                <element name="IntellectualEntityType " type="string"/>
                                <element name="hasRepresentation" minOccurs="0" maxOccurs="unbounded">
                                        <complexType>
                                                <attribute name="idref" type="IDREF" use="required"/>
```

```
                                        <!-- Representation ID -->
                                    </complexType>
                                </element>
                            </sequence>
                        </extension>
                    </complexContent>
</complexType>

<complexType name="RepresentationType">
                    <complexContent>
                        <extension base="pp2:PreservationObjectType">
                            <sequence>
                                <element name="embodiesIntellectualEntity" minOccurs="1" maxOccurs="unbounded">
                                    <complexType>
                                        <attribute name="idref" type="IDREF" use="required"/>
                                        <!-- IntellectualEntity ID -->
                                    </complexType>
                                </element>
                                <element name="hasRepresentationBitstream" minOccurs="1" maxOccurs="unbounded">
                                    <complexType>
                                        <attribute name="idref" type="IDREF" use="required"/>
                                        <!-- RepresentationBitstream ID -->
                                    </complexType>
                                </element>
                                <element name="hasRepresentationBitstreamStructmap" minOccurs="1"
                                    maxOccurs="unbounded">
                                    <!-- This should be like a METS structmap. To be extended by xml-->
                                </element>
                            </sequence>
                        </extension>
                    </complexContent>
</complexType>
```

```
<complexType name="RepresentationBitstreamType">
	<sequence>
		<element name="hasRepresentation" maxOccurs="unbounded" minOccurs="1">
			<complexType>
				<attribute name="idref" type="IDREF" use="required"/>
				<!-- Representation ID -->
			</complexType>
		</element>
		<element name="implementedBy" minOccurs="0" maxOccurs="1">
			<complexType>
				<attribute name="idref" type="IDREF" use="required"/>
				<!-- Bitstream ID -->
				<!-- A relationship between Bitstream and RepresentationBitstream is mandatory in at least one direction -->
			</complexType>
		</element>
	</sequence>
</complexType>

<complexType name="BitstreamType">
	<complexContent>
		<extension base="pp2:PreservationObjectType">
			<sequence>
				<element name="implements" minOccurs="0" maxOccurs="unbounded">
					<!-- A relationship between Bitstream and RepresentationBitstream is mandatory in at least one direction -->
					<complexType>
						<attribute name="idref" type="IDREF" use="required"/>
						<!-- RepresentationBitstream ID -->
					</complexType>
				</element>
			</sequence>
		</extension>
```

```
        </complexContent>
</complexType>

<complexType name="BytestreamType">
        <complexContent>
                <extension base="pp2:BitstreamType"> </extension>
        </complexContent>
</complexType>

<complexType name="FileType">
        <complexContent>
                <extension base="pp2:BytestreamType"> </extension>
        </complexContent>
</complexType>

<complexType name="EnvironmentType">
        <sequence>
                <element name="environmentName" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <element name="environmentDescription" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <element name="environmentPurpose" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <!-- creation, ingest, preservation, remote access, local access, migration, etc. extensible controlled vocabulary-->
                <element name="environmentFunction" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <!-- rendering, editing, executing, printing, etc. extensible controlled vocabulary-->
                <element name="environmentIntention" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <!-- necessary, recommended, acceptable, etc. extensible controlled vocabulary-->
                <element name="hasRelatedEnvironment" maxOccurs="unbounded" minOccurs="0">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- Environment ID -->
                        </complexType>
                </element>
                <element name="isEnvironmentOfObject" maxOccurs="unbounded" minOccurs="0">
```

```
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- PreservationObject ID -->
        </complexType>
</element>
<element name="isEnvironmentOfAction" maxOccurs="unbounded" minOccurs="0">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- PreservationAction ID -->
        </complexType>
</element>
<element name="actsAsPreservationService" maxOccurs="unbounded" minOccurs="0">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- PreservationService ID -->
        </complexType>
</element>
<element name="hasCharacteristic" maxOccurs="unbounded" minOccurs="0">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- Characteristic ID -->
        </complexType>
</element>
<element name="hasRisk" maxOccurs="unbounded" minOccurs="0">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- PreservationRisk ID -->
        </complexType>
</element>
<element name="hasRight" minOccurs="0" maxOccurs="unbounded">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
```

```
                        <!-- Right ID -->
                    </complexType>
                </element>
                <element name="hasPolicy" minOccurs="0" maxOccurs="unbounded">
                    <complexType>
                        <attribute name="idref" type="IDREF" use="required"/>
                        <!-- Policy ID -->
                    </complexType>
                </element>
                <element name="hasEvent" maxOccurs="unbounded" minOccurs="0">
                    <complexType>
                        <attribute name="idref" type="IDREF" use="required"/>
                        <!-- Event ID -->
                    </complexType>
                </element>
        </sequence>
        <attribute name="id" type="ID" use="required"/>
        <!-- Environment ID -->
</complexType>

<complexType name="PreservationRiskType">
        <sequence>
                <element name="riskName" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <element name="riskDescription" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <element name="associatedWith" maxOccurs="unbounded" minOccurs="0">
                    <complexType>
                        <attribute name="idref" type="IDREF" use="required"/>
                        <!-- PreservationObject or Environment ID -->
                    </complexType>
                </element>
                <element name="hasEvent" maxOccurs="unbounded" minOccurs="0">
                    <complexType>
```

```
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- Event ID -->
                            </complexType>
                        </element>
                </sequence>
                <attribute name="id" type="ID" use="required"/>
                <!-- PreservationRisk ID -->
                <attribute name="hasConstraint" type="IDREF" use="required"/>
                <!-- RiskSpecifyiingConstraintID -->
</complexType>

<complexType name="NewVersionRiskType">
        <complexContent>
                <extension base="pp2:PreservationRiskType"> </extension>
        </complexContent>
</complexType>

<complexType name="DeteriorationOrLossRiskType">
        <complexContent>
                <extension base="pp2:PreservationRiskType"> </extension>
        </complexContent>
</complexType>

<complexType name="LackingSupportRiskType">
        <complexContent>
                <extension base="pp2:PreservationRiskType"> </extension>
        </complexContent>
</complexType>

<complexType name="ProprietaryRiskType">
        <complexContent>
                <extension base="pp2:PreservationRiskType"> </extension>
```

```
        </complexContent>
</complexType>

<complexType name="UnmanagedGrowthRiskType">
        <complexContent>
                <extension base="pp2:PreservationRiskType"> </extension>
        </complexContent>
</complexType>

<complexType name="PreservationActionType">
        <sequence>
                <element name="actionName" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <element name="actionDescription" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <element name="relatedTo" maxOccurs="unbounded" minOccurs="0">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- PreservationAction ID -->
                        </complexType>
                </element>
                <element name="Type" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="hasRisk" minOccurs="0" maxOccurs="unbounded">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- PreservationRisk ID -->
                        </complexType>
                </element>
                <!-- At least one input or output PreservationObject or Environment needs to exist -->
                <element name="hasInputPreservationObject" maxOccurs="1" minOccurs="0">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- PreservationObject ID -->
                        </complexType>
```

```
</element>
<element name="hasOutputPreservationObject" maxOccurs="1" minOccurs="0">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- PreservationObject ID -->
        </complexType>
</element>
<element name="hasInputEnvironment" maxOccurs="1" minOccurs="0">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- Environment ID -->
        </complexType>
</element>
<element name="hasOutputEnvironment" maxOccurs="1" minOccurs="0">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- Environment ID -->
        </complexType>
</element>
<element name="hasEnvironment" minOccurs="0" maxOccurs="unbounded">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- Environment ID -->
        </complexType>
</element>
<element name="hasPreservationService" minOccurs="0" maxOccurs="unbounded">
        <complexType>
                <attribute name="idref" type="IDREF" use="required"/>
                <!-- PreservationService ID -->
        </complexType>
</element>
<element name="hasCharacteristic" minOccurs="0" maxOccurs="unbounded">
```

```
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- Characteristic ID -->
                        </complexType>
                </element>
                <element name="hasEventOutcome" minOccurs="0" maxOccurs="1">
                        <complexType>
                                <sequence>
                                        <element name="hasPolicy" minOccurs="0" maxOccurs="unbounded">
                                                <complexType>
                                                        <attribute name="idref" type="IDREF" use="required"/>
                                                        <!-- Policy -->
                                                </complexType>
                                        </element>
                                        <element name="DegreeOfCompliance" minOccurs="0" maxOccurs="unbounded">
                                                <complexType>
                                                        <sequence>
                                                                <element name="associatedWith" minOccurs="1" maxOccurs="unbounded"
type="IDREF"/>
                                                                <!-- a vector of parameters which are either PreservationObject or Environment IDs -->
                                                        </sequence>
                                                        <attribute name="hasConstraint" type="IDREF" use="required"/>
                                                        <attribute name="hasMeasure" type="IDREF" use="required"/>
                                                        <!-- The definition of measure is out-of-scope for tis model -->
                                                        <attribute name="hasOutcome" type="string" use="required"/>
                                                </complexType>
                                        </element>
                                </sequence>
                        </complexType>
                </element>
        </sequence>
        <attribute name="id" type="ID" use="required"/>
```

```
        <!-- PreservationAction ID -->
</complexType>

<complexType name="PropertyType">
        <sequence>
                <element name="propertyName" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <element name="propertyDescription" maxOccurs="unbounded" minOccurs="0" type="string"/>
                <element name="appliesTo" maxOccurs="unbounded" minOccurs="1">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- PreservationObject or Environment or PreservationAction ID -->
                        </complexType>
                </element>
                <element name="hasRange" maxOccurs="unbounded" minOccurs="0">
                        <complexType>
                                <attribute name="hasUnit" type="IDREF" use="optional"/>
                                <!-- Unit ID -->
                                <attribute name="hasDataConstraint" type="string" use="required"/>
                                <attribute name="isDefault" type="string" use="required"/>
                                <!-- yes, no -->
                                <attribute name="hasDefaultValue" type="string" use="required"/>
                                <!-- This could be any value, number, string, etc.-->
                        </complexType>
                </element>
                <element name="hasValueOrigin" maxOccurs="unbounded" minOccurs="0">
                        <complexType>
                                <attribute name="hasValueOriginID" type="string" use="required"/>
                                <!-- ValueOrigin ID -->
                                <attribute name="isDefault" type="IDREF" use="optional"/>
                                <!-- yes, no -->
                        </complexType>
                </element>
```

```
                <element name="hasRelationship" minOccurs="0" maxOccurs="unbounded">
                        <complexType>
                                <attribute name="hasRelatedProperty" type="IDREF" use="required"/>
                                <!-- Property ID -->
                                <attribute name="hasRelationshipType" type="string" use="required"/>
                        </complexType>
                </element>
                <element name="hasEvent" minOccurs="0" maxOccurs="unbounded">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- Event ID -->
                        </complexType>
                </element>
        </sequence>
        <attribute name="id" type="ID" use="required"/>
        <!-- Property ID -->
</complexType>

<complexType name="ValueOriginType">
        <sequence>
                <element name="valueOriginName" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="valueOriginDescription" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="hasSource" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="hasTargetUnit" minOccurs="0" maxOccurs="unbounded" type="IDREF"/>
                <!-- Unit ID -->
                <element name="hasTechnique" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="hasAgent" minOccurs="0" maxOccurs="unbounded" type="IDREF"/>
                <!-- Agent ID -->
                <element name="hasTrigger" minOccurs="0" maxOccurs="unbounded" type="string"/>
        </sequence>
        <attribute name="id" type="ID" use="required"/>
        <!-- ValueOrigin ID -->
```

```
</complexType>

<complexType name="UnitType">
        <sequence>
                <element name="unitName" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="unitDescription" minOccurs="0" maxOccurs="unbounded" type="string"/>
                <element name="hasDataConstraint" minOccurs="1" maxOccurs="1" type="string"/>
                <element name="hasConversion" minOccurs="0" maxOccurs="unbounded">
                        <complexType>
                                <sequence>
                                        <element name="hasTechnique" minOccurs="0" maxOccurs="unbounded"
                                                type="string"/>
                                        <element name="hasAgent" minOccurs="0" maxOccurs="unbounded" type="IDREF"/>
                                        <!-- Agent ID -->
                                </sequence>
                                <attribute name="hasSource" type="IDREF" use="required"/>
                                <!-- Unit ID -->
                        </complexType>
                </element>
        </sequence>
        <attribute name="id" type="ID" use="required"/>
        <!-- Unit ID -->
</complexType>

<complexType name="CharacteristicType">
        <sequence>
                <element name="associatedWith" maxOccurs="unbounded" minOccurs="1">
                        <complexType>
                                <attribute name="idref" type="IDREF" use="required"/>
                                <!-- PreservationObject or Environment or PreservationAction ID -->
                        </complexType>
                </element>
```

```
                <!-- a vector of parameters which are either PreservationObject or Environment or PreservationAction IDs -->
                <element name="Annotation" minOccurs="0" maxOccurs="1">
                        <complexType>
                                <attribute name="hasUnit" type="IDREF" use="optional"/>
                                <!-- Unit ID -->
                                <attribute name="hasValueOrigin" type="IDREF" use="optional"/>
                                <!-- ValueOrigin ID -->
                                <attribute name="hasTechnique" type="string" use="optional"/>
                                <attribute name="hasSource" type="string" use="optional"/>
                                <attribute name="hasAgent" type="IDREF" use="optional"/>
                                <!-- Agent ID -->
                        </complexType>
                </element>
        </sequence>
        <attribute name="id" type="ID" use="required"/>
        <!-- Characteristic ID -->
        <attribute name="hasProperty" type="IDREF" use="required"/>
        <!-- Property ID -->
        <attribute name="onDemand" type="string" use="optional"/>
        <!-- yes, no -->
        <attribute name="hasValue" type="string" use="optional"/>
        <attribute name="hasCreationEvent" type="IDREF" use="optional"/>
        <!-- Event ID -->
</complexType>

<complexType name="ConstraintType">
        <sequence>
                <element name=" constraintName" type="string" maxOccurs="unbounded" minOccurs="0"/>
                <element name=" constraintDescription" type="string" maxOccurs="unbounded" minOccurs="0"/>
                <element name="hasPolicy" type="IDREF" maxOccurs="unbounded" minOccurs="0"/>
                <!-- Policy ID -->
                <element name="hasStakeholder" type="IDREF" maxOccurs="unbounded" minOccurs="0"/>
```

```
                    <!-- Agent ID -->
                    <element name=" constraintApplicability" maxOccurs="1" minOccurs="0">
                        <complexType>
                            <attribute name="startDate" type="date" use="optional"/>
                            <attribute name="endDate" type="date" use="optional"/>
                        </complexType>
                    </element>
                    <element name="Specification" maxOccurs="1" minOccurs="1">
                        <complexType>
                        <sequence>
                            <element name="context" minOccurs="0" maxOccurs="unbounded" type="string"/>
                            <element name="precondition" minOccurs="0" maxOccurs="unbounded" type="string"/>
                            <element name="postcondition" minOccurs="0" maxOccurs="unbounded" type="string"/>
                            <!-- The postcondition includes the tolerance factor -->
                            <!-- No necessity to invent an XML constraint language at this juncture. Type "string" allows for versatile use. For an
example UML constraint language see OCL (the Object Constraint Language) -->
                        </sequence>
                        </complexType>
                    </element>
                    <element name=" constraintImportanceFactor" type="string" maxOccurs="1" minOccurs="0"/>
                    <element name="hasEvent" type="IDREF" maxOccurs="unbounded" minOccurs="0"/>
                    <!-- Event ID -->
                </sequence>
                <attribute name="id" type="ID" use="required"/>
                <!-- Constraint ID -->
            </complexType>

            <complexType name="NonPreservationConstraintType">
                <complexContent>
                    <extension base="pp2:ConstraintType"> </extension>
                </complexContent>
            </complexType>
```

```
<complexType name="PreservationConstraintType">
        <complexContent>
                <extension base="pp2:ConstraintType"> </extension>
        </complexContent>
</complexType>

<complexType name="RiskSpecifyingConstraintType">
        <complexContent>
                <extension base="pp2:PreservationConstraintType"> </extension>
        </complexContent>
</complexType>

<complexType name="PreservationObjectSelectingConstraintType">
        <complexContent>
                <extension base="pp2:RiskSpecifyingConstraintType"> </extension>
        </complexContent>
</complexType>

<complexType name="PreservationProcessGuidingConstraintType">
        <complexContent>
                <extension base="pp2:PreservationConstraintType"> </extension>
        </complexContent>
</complexType>

<complexType name="PreservationInfrastructureConstraintType">
        <complexContent>
                <extension base="pp2:PreservationProcessGuidingConstraintType"> </extension>
        </complexContent>
</complexType>

<complexType name="PreservationGuidingConstraintType">
```

```
        <complexContent>
                <extension base="pp2:PreservationConstraintType"> </extension>
        </complexContent>
</complexType>

<complexType name="ActionDefiningConstraintType">
        <complexContent>
                <extension base="pp2:PreservationGuidingConstraintType"> </extension>
        </complexContent>
</complexType>

<complexType name="SignificanceConstraintType">
        <complexContent>
                <extension base="pp2:PreservationGuidingConstraintType"> </extension>
        </complexContent>
</complexType>

<complexType name="RiskActionMatchingConstraintType">
        <complexContent>
                <extension base="pp2:PreservationGuidingConstraintType"> </extension>
        </complexContent>
</complexType>

<element name="ActionDefiningConstraint" type="pp2:ActionDefiningConstraintType"/>
<element name="Bitstream" type="pp2:BitstreamType"/>
<element name="Bytestream" type="pp2:BytestreamType"/>
<element name="Characteristic" type="pp2:CharacteristicType"/>
<element name="DeteriorationOrLossRisk" type="pp2:DeteriorationOrLossRiskType"/>
<element name="Environment" type="pp2:EnvironmentType"/>
<element name="File" type="pp2:FileType"/>
<element name="IntellectualEntity" type="pp2:IntellectualEntityType"/>
<element name="LackingSupportRisk" type="pp2:LackingSupportRiskType"/>
```

```
    <element name="NewVersionRisk" type="pp2:NewVersionRiskType"/>
    <element name="NonPreservationConstraint" type="pp2:NonPreservationConstraintType"/>
    <element name="PreservationAction" type="pp2:PreservationActionType"/>
    <element name="PreservationGuidingConstraint" type="pp2:PreservationGuidingConstraintType"/>
    <element name="Policy"
             type="pp2:PolicyType"/>
    <element name="PreservationInfrastructureConstraint"
             type="pp2:PreservationInfrastructureConstraintType"/>
    <element name="PreservationObject" type="pp2:PreservationObjectType"/>
    <element name="PreservationObjectSelectingConstraint"
             type="pp2:PreservationObjectSelectingConstraintType"/>
    <element name="PreservationProcessGuidingConstraint"
             type="pp2:PreservationProcessGuidingConstraintType"/>
    <element name="PreservationConstraint" type="pp2:PreservationConstraintType"/>
    <element name="PreservationRisk" type="pp2:PreservationRiskType"/>
    <element name="Property" type="pp2:PropertyType"/>
    <element name="ProprietaryRisk" type="pp2:ProprietaryRiskType"/>
    <element name="RepresentationBitstream" type="pp2:RepresentationBitstreamType"/>
    <element name="Representation" type="pp2:RepresentationType"/>
    <element name="Constraint" type="pp2:ConstraintType"/>
    <element name="RiskActionMatchingConstraint" type="pp2:RiskActionMatchingConstraintType"/>
    <element name="RiskSpecifyingConstraint" type="pp2:RiskSpecifyingConstraintType"/>
    <element name="SignificanceConstraint"
             type="pp2:SignificanceConstraintType"/>
    <element name="Unit" type="pp2:UnitType"/>
    <element name="UnmanagedGrowthRisk" type="pp2:UnmanagedGrowthRiskType"/>
    <element name="ValueOrigin" type="pp2:ValueOriginType"/>

</schema>
```

## 7.3 Example scenario for the DePICT modelling approach

### 7.3.1 Example scenario

A museum has a collection of hand-drawn maps. They are creating digital copies for easier access for the public, in order to protect the originals and for insurance purposes. The digitised images are to be treated as preservation master copies. Access copies will be derived from them. The museum outsources the digitisation and receives the digitised images as high-resolution camera raw images as .dng files in the Adobe Digital Negative Raw Image file format.

The museum has a written digital preservation policy that applies to this collection. It states a set of constraints that need to be satisfied in order to ensure the long-term preservation of its collection items. The curator checks against all stated constraints in this policy. The output of this activity is an assessment of the presence and severity of preservation risks. One constraint states that all high-value preservation masters for image files must have the associated camera profile information. She runs the C3PO collection profiling software (TU Wien, nd-b) to identify the images in the files received. Upon visual inspection of the files she finds that there is no camera profile information included in the .dng metadata and there is no separate technical metadata. This means that the risk specifying constraint has been violated.

Since this is a “must” requirement the curator needs to plan mitigating actions and decide on the best choice of preservation action. The first step is to obtain the necessary information to create a camera profile. The digital curator contacts the digitisation supplier and requests camera profile information. The supplier sends a camera profile file in .dcp format.

In a pre-selection step she identifies several preservation service options for mitigating the risk so that the constraint is satisfied.

1) She can extract the camera profile information from the .dcp and embed it directly into the .dng metadata.
    a) She can use her Lightroom software (Adobe, nd) to change the camera profile information of the images through the software user interface.
    b) She can write a batch script.
2) She can keep the camera profile information separately as .dcp file.
    a) She can bundle it with the images using a container format, such as .zip.

b) She can link the camera profile file and each image file in a metadata container, such as METS, by creating a logical link in the structural metadata.

This time she uses preservation-guiding constraints from the preservation policy to help her choose the right transformation preservation action. There is a preservation-guiding constraint that states that technical metadata that is necessary for proper rendering “should” be embedded in the image file. This means that Option 1 is preferable. There is also a common-sense constraint that tells her that the solution effort has to be proportional to the task. Since the number of images is small enough they can be easily processed using the user interface of Lightroom; the effort of manually embedding the profile in a small number of preservation objects is not prohibitive. A script alternative would require more time and the script is not anticipated to become part of the standard workflow. It does not seem to her that the effort would pay off in the long term. She decides on Option 1a and once more validates that this option best matches the preservation-guiding constraints and successfully mitigates the risk-specifying constraint. There are also significance constraints that need to be evaluated. They might include constraints that assure that the metadata was embedded correctly as specified in the .dcp profile, that no other metadata has been lost in the process, and that the image content is unchanged. She will also verify that the number of images transformed is equal to the number of images received from the digitisation supplier. In order to validate that the metadata was embedded correctly as specified in the .dcp profile she needs to define the relationship between the metadata elements in the .dcp file and the resulting .dng file. The preservation must be successful against this definition of equality. In the case of .dcp and .dng files, the corresponding metadata fields have identical names and the mapping is straight-forward. Alternatively, she could obtain a guarantee from the testbed (discussed in section 4.2), which establishes the preservation properties of preservation execution services. The testbed might have already established the fact that the Lightroom software correctly embeds camera profile metadata for the type of images she uses.

Now she can execute the chosen transformation preservation action. For quality assurance purposes, she needs to validate that the constraints have been satisfied. The output of the validation action is a measure of the degree of compliance of the preservation action with the set of constraints.

She records the constraints that guided her decisions and the event details of the transformation preservation action as provenance metadata and stores them together with the images in the repository.

### *7.3.2 Key entities in the scenario*

DePICT, as a conceptual model, can be used to clarify scenarios in the domain. In this section the scenario is analysed in order to identify the corresponding DePICT entities. The key entities in this scenario can be categorised according to the class model in Figure 45 and are listed in Table 6. DePICT is not a data model. Not every identified concept will need to be recorded as metadata or implemented through supporting preservation software. A requirements analysis determines which metadata needs to be recorded or which functions need to be supported through manual or automatic workflow steps.

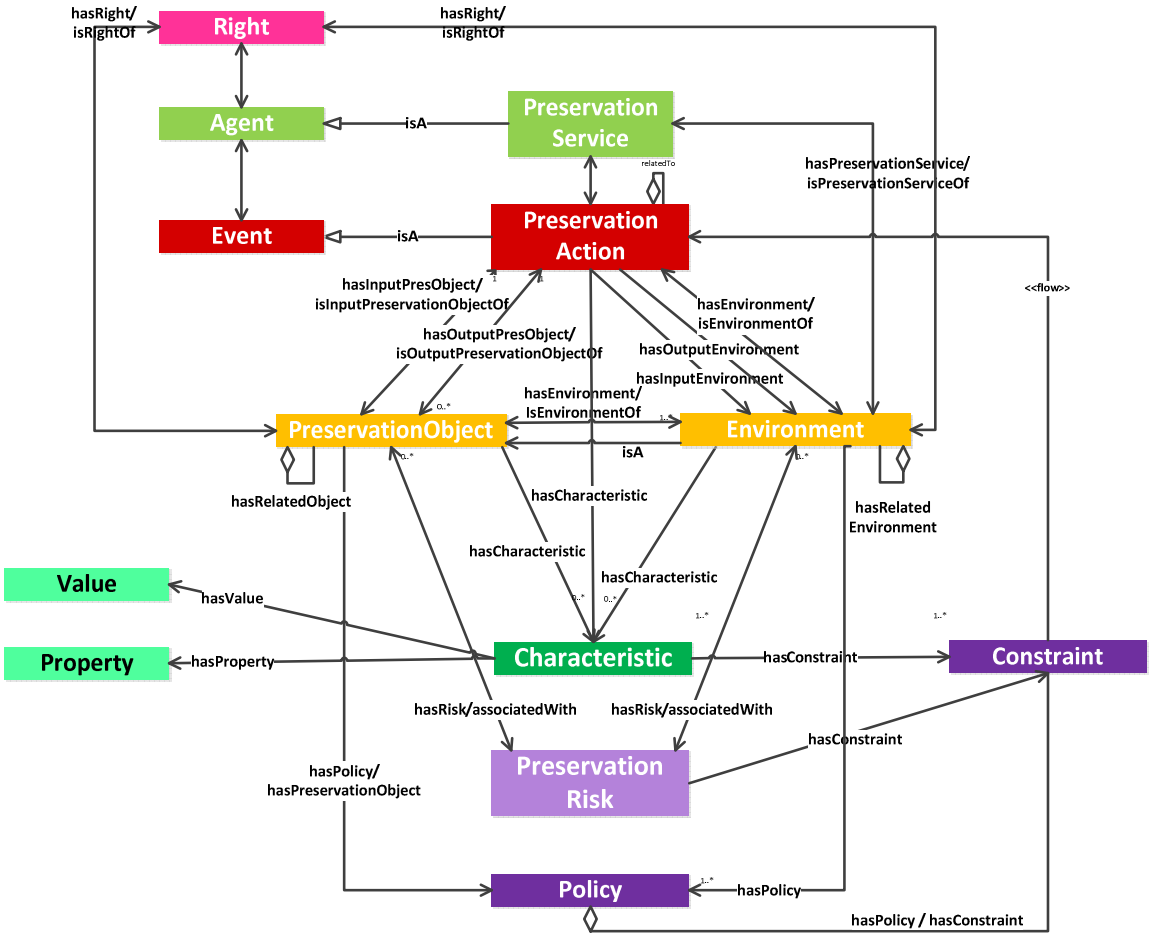



**Figure 45: Full conceptual model**

**Table 6: DePICT entities in the example scenario**

| *Entity type* | Entities in the scenario | Figure |
|---|---|---|
| *Agent* | • Museum<br>• Digital curator<br>• Digitisation supplier | See Figure 45 |
| *Agent / PreservationService* | • C3PO<br>• Testbed<br>• Lightroom software<br>• Script for adding camera profile to .dng file<br>• Script for adding camera profile to a METS container<br>• Script for adding camera profile to a.zip file | |
| *Environment* | • Camera<br>• Lightroom software<br>• C3PO tool<br>• Container format for bundling metadata and files (METS, zip)<br>• File format specification for .dng<br>• File format specification for .dcp<br>• Hex-editor | See Figure 45 |
| *PreservationObject* | • Collection of hand-drawn maps<br>• Digitised high-resolution camera raw images as .dng files without camera profile embedded<br>• Digitised high-resolution camera raw images as .dng files with camera profile embedded<br>• Access copies<br>• Camera profile file in .dcp format | See Figure 45 |
| *PreservationObjects / Component* | • .dng file metadata fields<br>• .dcp file metadata fields | |
| *Characteristics* | • The number of images is small<br>• Lightroom embeds camera profile metadata.<br>• An alternative script is not anticipated to become part of the standard workflow | See Figure 45 |

| *Entity type* | Entities in the scenario | Figure |
|---|---|---|
| *Properties* | • numberOfImages (file set): integer<br>• embedsCameraProfileMetadata (software): binary<br>• partOfStandardWorkflow (service): binary<br>• BaselineExposureOffset<br>• CalibrationIlluminant1<br>• CalibrationIlluminant2<br>• ColorMatrix1<br>etc. | See Figure 45 |
| *Policy* | • Digital preservation policy that applies to the collection of hand-drawn maps<br>• Common sense<br>• Constraints derived for the specific situation | See Figure 45 |
| *Constraint/ RiskSpecifying-Constraint* | • All high-value preservation masters for image files must have the associated camera profile information. “must” | See Figure 46 |
| *Constraint/ PreservationObject-SelectingConstraint* | • Preservation objects that have file formats taken from the image format list in the policy document are considered image files. “must”<br>This constraint helps to identify the files that are affected by the risk specifying constraint. | |
| *Constraint/ PreservationGuiding-Constraint* | • Technical metadata that is necessary for proper rendering should be embedded in the image file. “should” | |
| *Constraint/ RiskActionMatching-Constraint* | • The chosen transformation preservation action must embed camera profile information in each output image. “must” | |
| *Constraint/ ActionDefining-Constraint* | • The solution effort should be proportional to the task. “should”<br>• The number of images transformed must be equal to the number of images received from the digitisation supplier. “must” | |

| *Entity type* | Entities in the scenario | Figure |
|---|---|---|
| *Constraint/ Significance-Constraint* | • Embedded camera profile metadata must be the same as the metadata specified in the .dcp profile. “must”<br>• Pre-existing metadata must be preserved in the process. “must”<br>• The image bit sequence must remain unchanged. “must” | See Figure 46 |
| *PreservationRisk* | • Images received from the supplier do not contain camera profile information. | See Figure 45 |

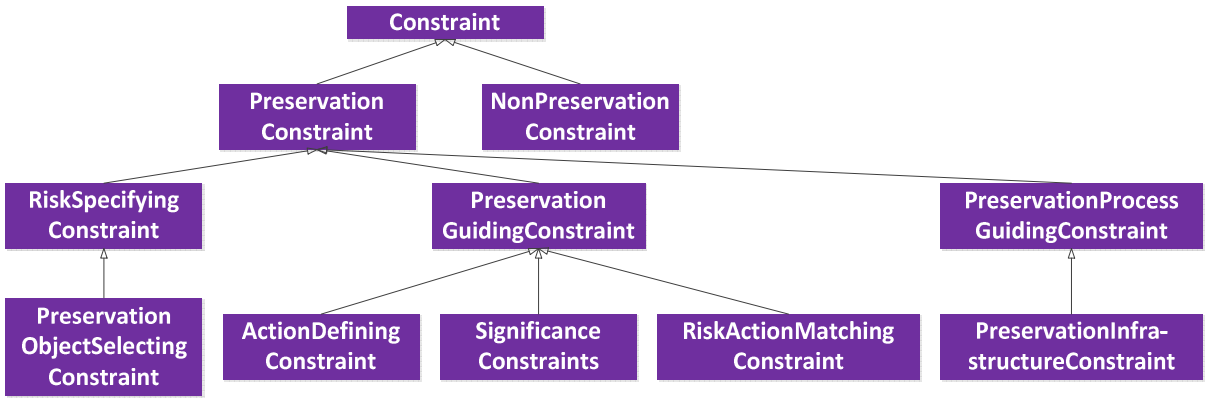



**Figure 46: Constraints**

| *Entity type* | Entities in the scenario | Figure |
|---|---|---|
| *Events/ PresevationActions/ PropertyDefinition* | • Identify properties used in the constraints (above) and their applicable vocabulary. | See Figure 47 |
| *Events/ PresevationActions/ BusinessModelling* | • Articulate constraints (above). | |
| *Events/ PresevationActions/ Characterisation* | • Identify image files in the set of files received from the supplier with CP3O.<br>• Identify technical metadata values in the image files with hex-editor or Lightroom. | |

| *Entity type* | Entities in the scenario | Figure |
|---|---|---|
| *Events/ PresevationActions/ Testbed Characterisation* | • Validate that the Lightroom software correctly embeds camera profile metadata. | |
| *Events/ PresevationActions/ Monitoring* | • Monitor the collection against the risk specifying constraints (above). | |
| *Events/ PresevationActions/ Preservation Planning* | • Evaluate which transformation preservation action candidate best satisfies the set of constraints above. | |
| *Events/ PresevationActions/ Transformation-PreservationAction* | • Obtain camera profile.<br>• Associate camera profile with original .dng's.<br>  ○ Embed in Lightroom.<br>  ○ Embed using script.<br>  ○ Bundle with .zip.<br>  ○ Bundle with METS. | |
| *Events/ PresevationActions/ TransformationPreservationAction Validation* | • Establish to what degree the constraints have been satisfied in the transformation preservation action. | |
| *Events/ PresevationActions/ MetadataStorage* | • Record the constraints that guided the decisions and the event details of the transformation preservation action as provenance metadata. | |
| *Events/ PresevationActions/ PreservationObject Storage* | • Store the information packages consisting of the .dng files, which contain content and camera profile metadata, as well as other preservation description metadata in the repository. | |

The steps described in the scenario in section 7.3.1 and under *Events / Presevation-Actions* in Table 6 can be mapped to the information exchange model for the digital preservation life cycle depicted in Figure 47.

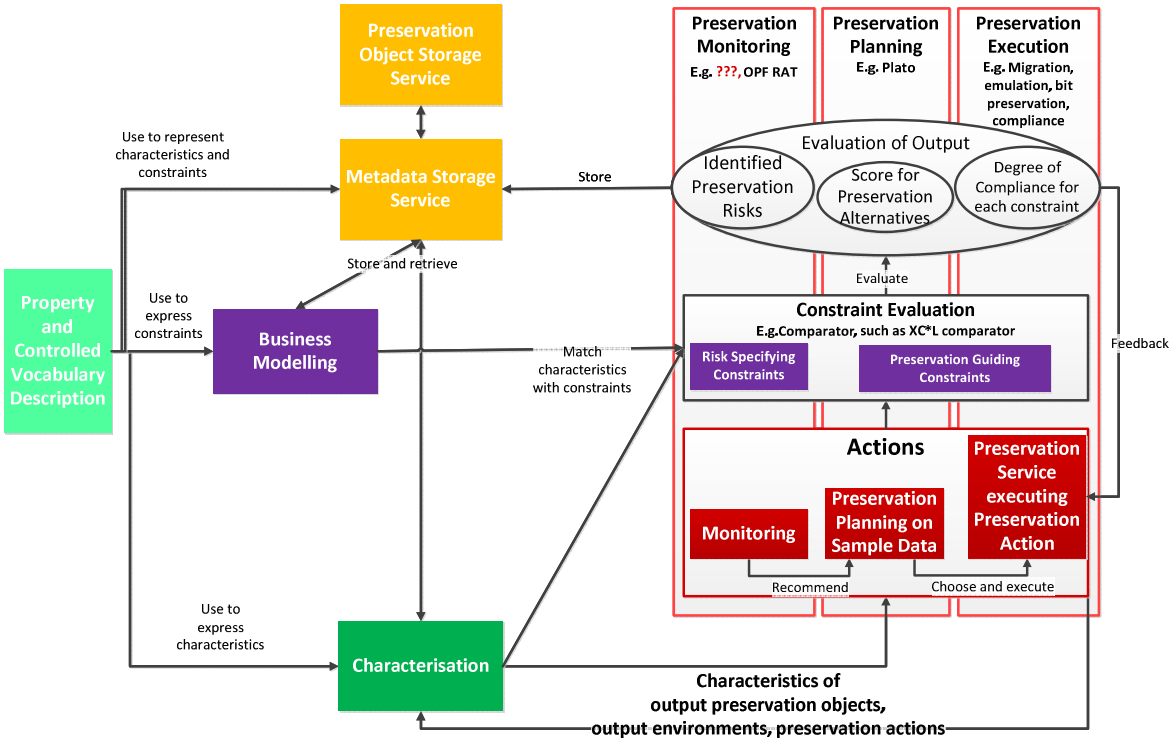



**Figure 47: Information exchange model for the digital preservation life cycle**

### *7.3.4 Key relationships in the scenario*

Section 7.3.2 identified the key entities in the scenario. This section analyses the relationships between them. Some examples of relationships between entities identified in Table 6 are depicted in Figures 48 ff. Figure 48 is an overview diagram which helps locate the following four vignettes.

- Figure 49 shows a first vignette, the *PreservationAction* flow. The *TransformationPreservationAction* "Embed Lightroom Metadata" is shown as part of the overall flow of *PreservationAction Events* and is discussed in more detail in the following.
- Vignette 2 in Figure 50 shows the relationship between the *TransformationPreservationAction* "Embed Lightroom Metadata" and its related *PreservationService* provided by a Lightroom installation. One of its *Characteristics* is ”*partOfStandardWorkflow* = yes”. Furthermore it has an *hasInputPreservationObject* relationship to the .dng file that is missing the camera profile metadata and the .dcp camera profile file and an *hasOutputPreservationObject* relationship to the .dng file with camera profile metadata. The Lightroom *Environment* is the *Environment* in which the *TransformationPreservationAction* is executed. It happens to also be the rendering *Environment* for the input and output *PreservationObjects*.
- Figure 51, Vignette 3, shows the *Constraints* in the scenario. The top *Constraint* is the *RiskSpecifyingConstraint* which states that all high-value preservation masters for image files must have the associated camera profile information. Because the *Property embedsCameraProfileMetadata* has the *Value* “no” for the *PreservationObject* the *Constraint* is violated by this *PreservationObject*. The violation of the *Constraint* now leads to the creation of an instantiated *PreservationRisk* for the *PreservationObject* which triggers the need for a *TransformationPreservationAction*. All *Constraints* guide the choice of *PreservationActions*. *Constraints* from multiple *Policies* apply to the input *PreservationObjects* : the explicit digital preservation policy, *Constraints* that need to be defined and satisfied for the specific situation and common sense *Constraints* .
- Vignette 4 in Figure 52 shows *PreservationObjects* and *Environments* involved in the *TransformationPreservationAction* in the example scenario. *PreservationObjects* in the scenario are managed on various levels. The *IntellectualEntity* on the top level is the collection of hand-drawn maps. It has an aggregate relationship to each

individual map in the collection, which again is described and managed as an abstract object, an *IntellectualEntity*. In the scenario, the "hand-drawn map" *IntellectualEntity* has three *Representations*:

- The one made up of the .dng file received initially, augmented by the later received .dcp camera profile file – the input files to the *TransformationPreservation-Action*,
- the one created after the *TransformationPreservationAction* which has the camera profile embedded in the .dng file's header,
- an access copy, which is not depicted in the diagrams.

One could further choose to model the metadata header fields in each file as *IntellectualEntity* of their own since they are managed separately and they play an important role for the *PreservationObjects'* use. The *RepresentationBitstream* is depicted for completeness sake, but not discussed in the scenario. The *Bitstreams* are the files associated with each *Representation*. The *TransformationPreservation-Action* has a *hasInputPreservationObject* relationship with the files in the first *Representation* and a *hasOutputPreservationObject* relationship with the files in the second *Representation*. It also has three *Environment* relationships with the Lightroom *Environment*, since it happens to be the *Environment* in which the *TransformationPreservationAction* is executed as well as the *Environment* in which the *InputPreservationObjects* and *OutputPreservationObject* are rendered. Vignette 4 furthermore shows some *Environments* that are relevant for the files in the scenario and whose *Characteristics* may present important representation information for the scenario: the camera and the file format specifications for the .dng and .dcp files. The figure also shows some selected *Properties* and *Characteristics* that apply to different entity types.

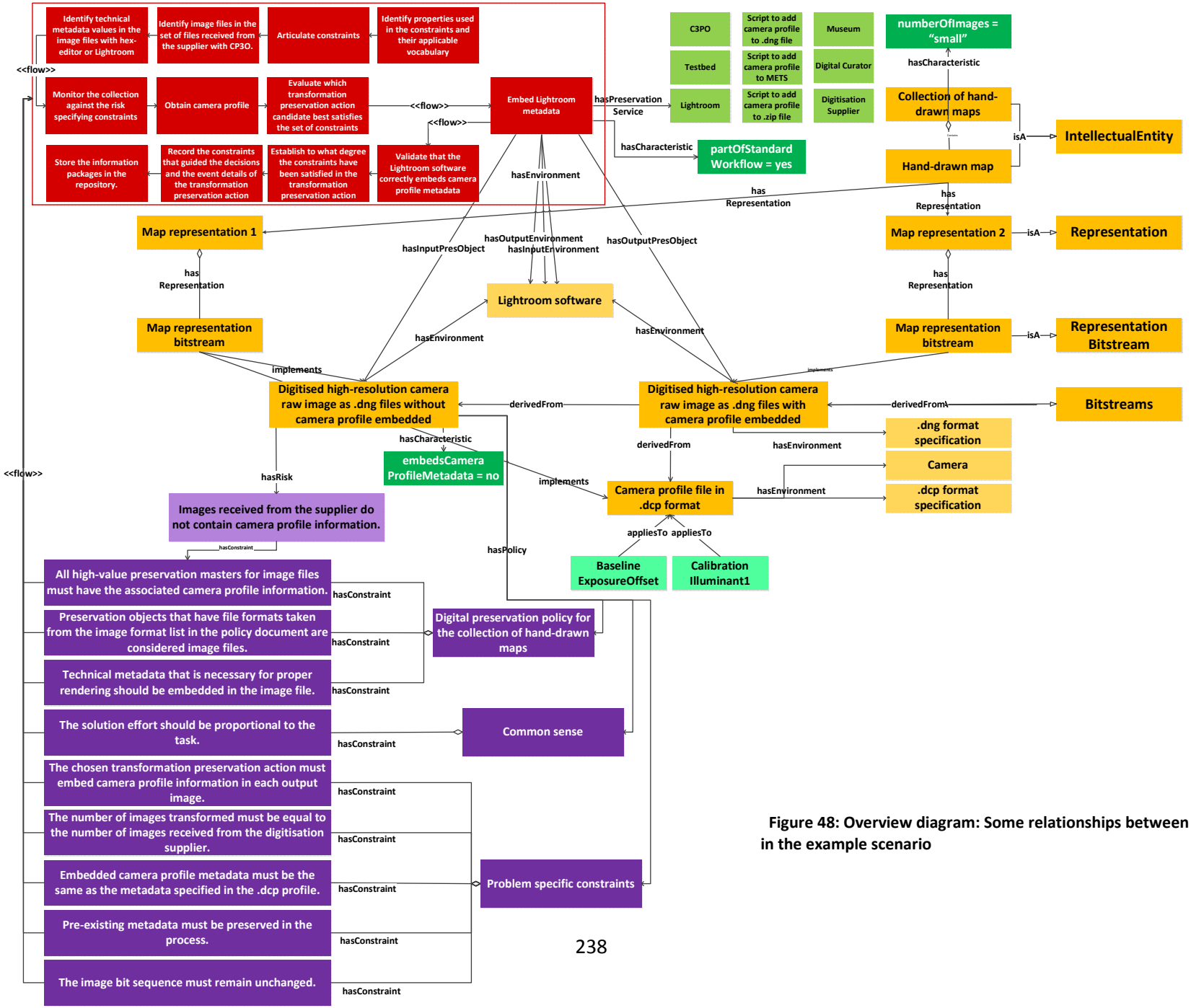
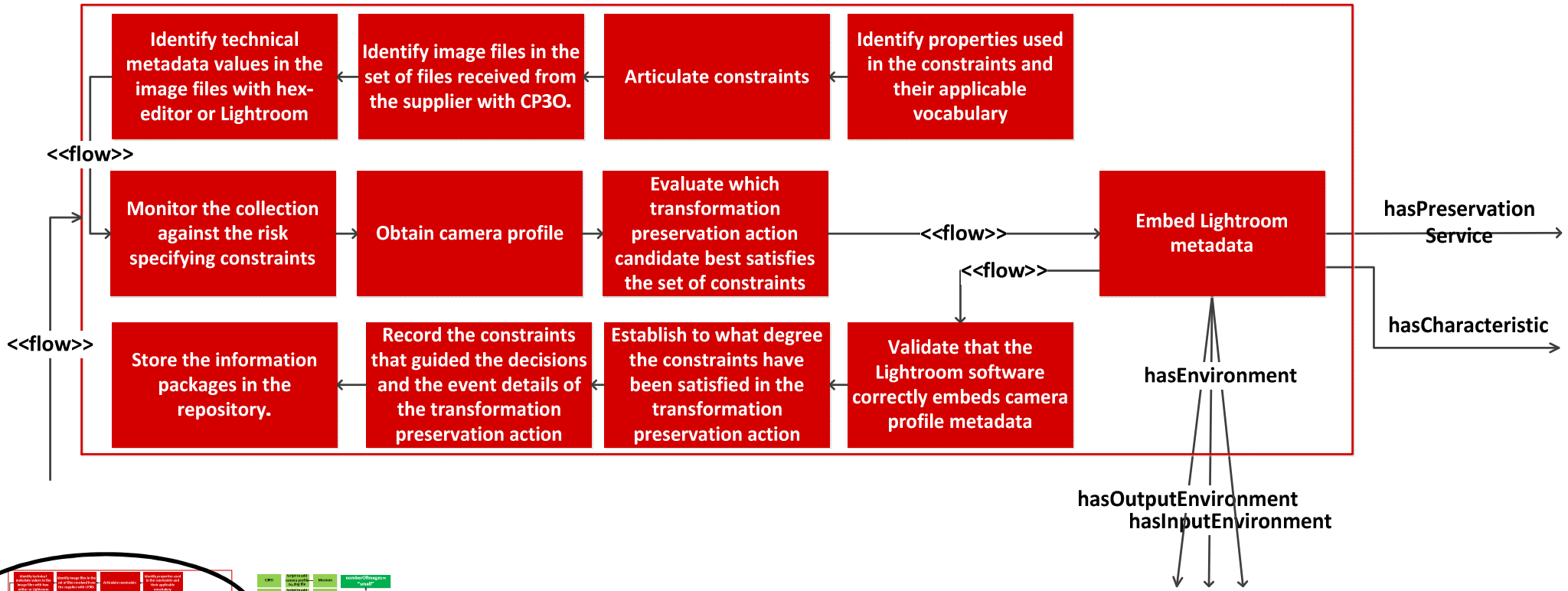



**Figure 48: Overview diagram: Some relationships between the entities in the example scenario**



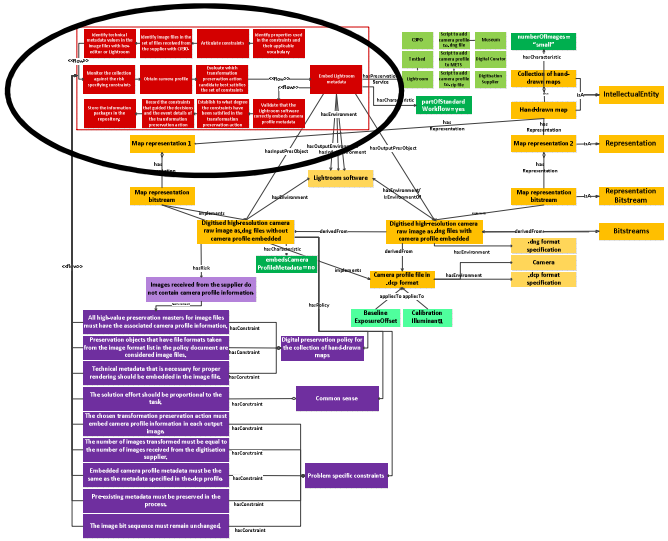
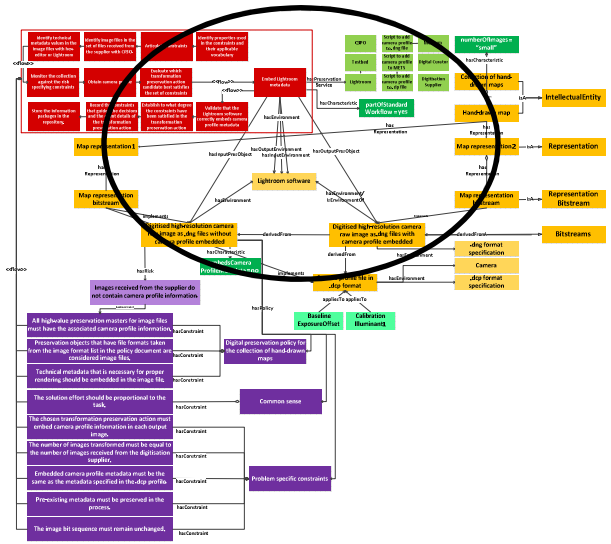



**Figure 49: Vignette 1: The `TransformationPreservationAction` "Embed Lightroom Metadata" as part of the overall `PreservationAction` flow of the example scenario.**



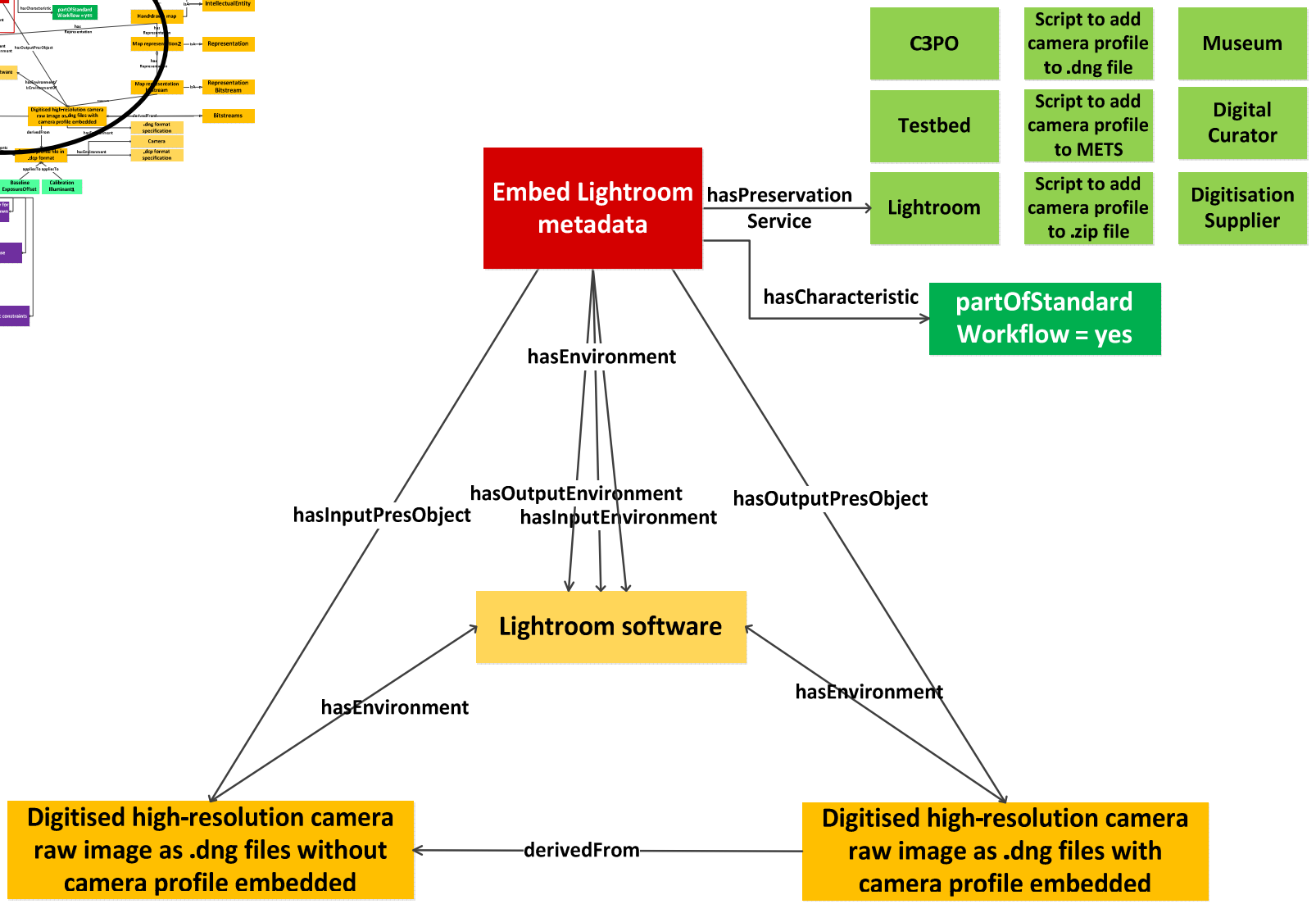



**Figure 50: Vignette 2: The relationships between the *`TransformationPreservationAction`* "Embed Lightroom Metadata" and its related entities in the example scenario.**

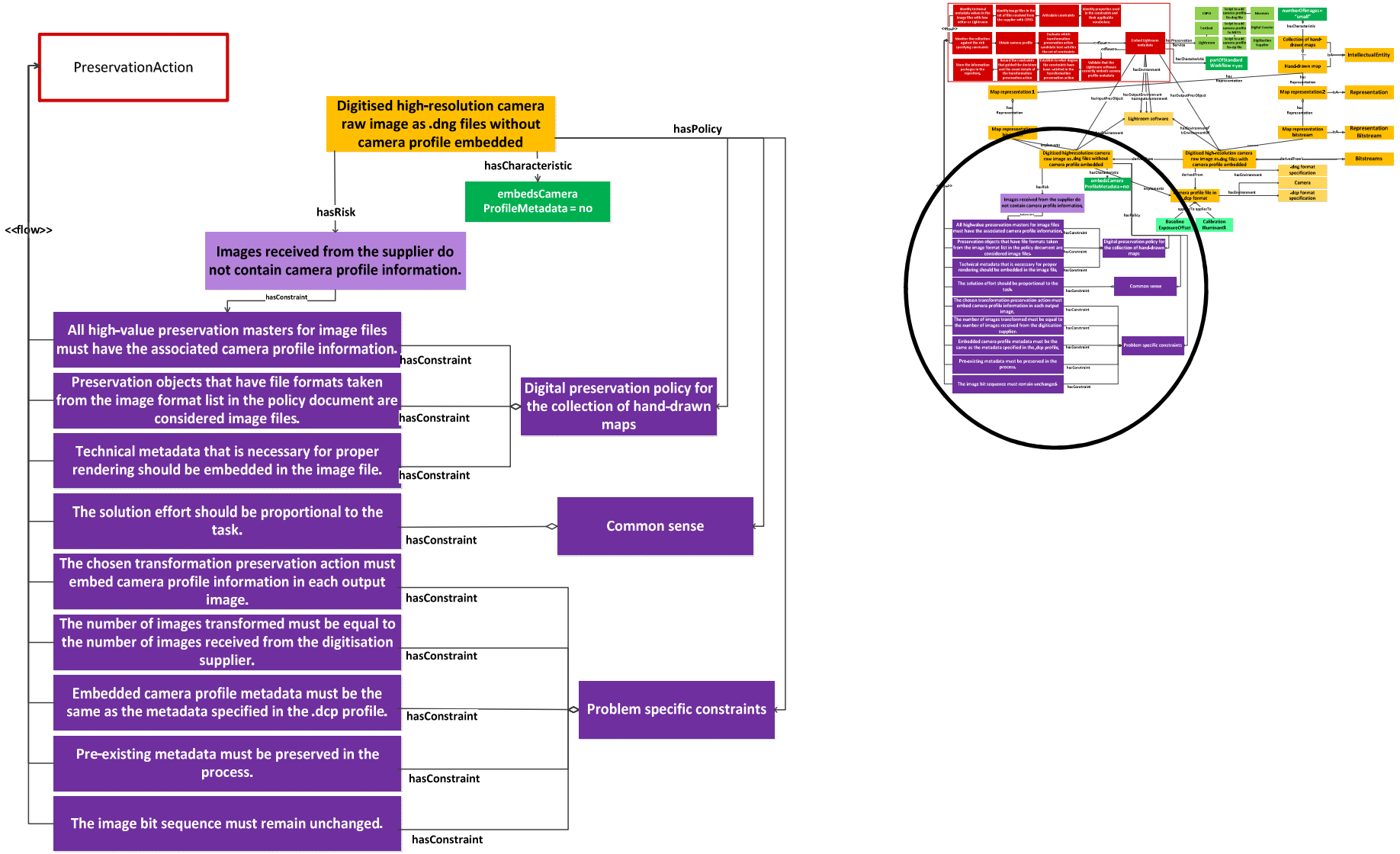



**Figure 51: Vignette 3: The *Constraints* in the scenario.**

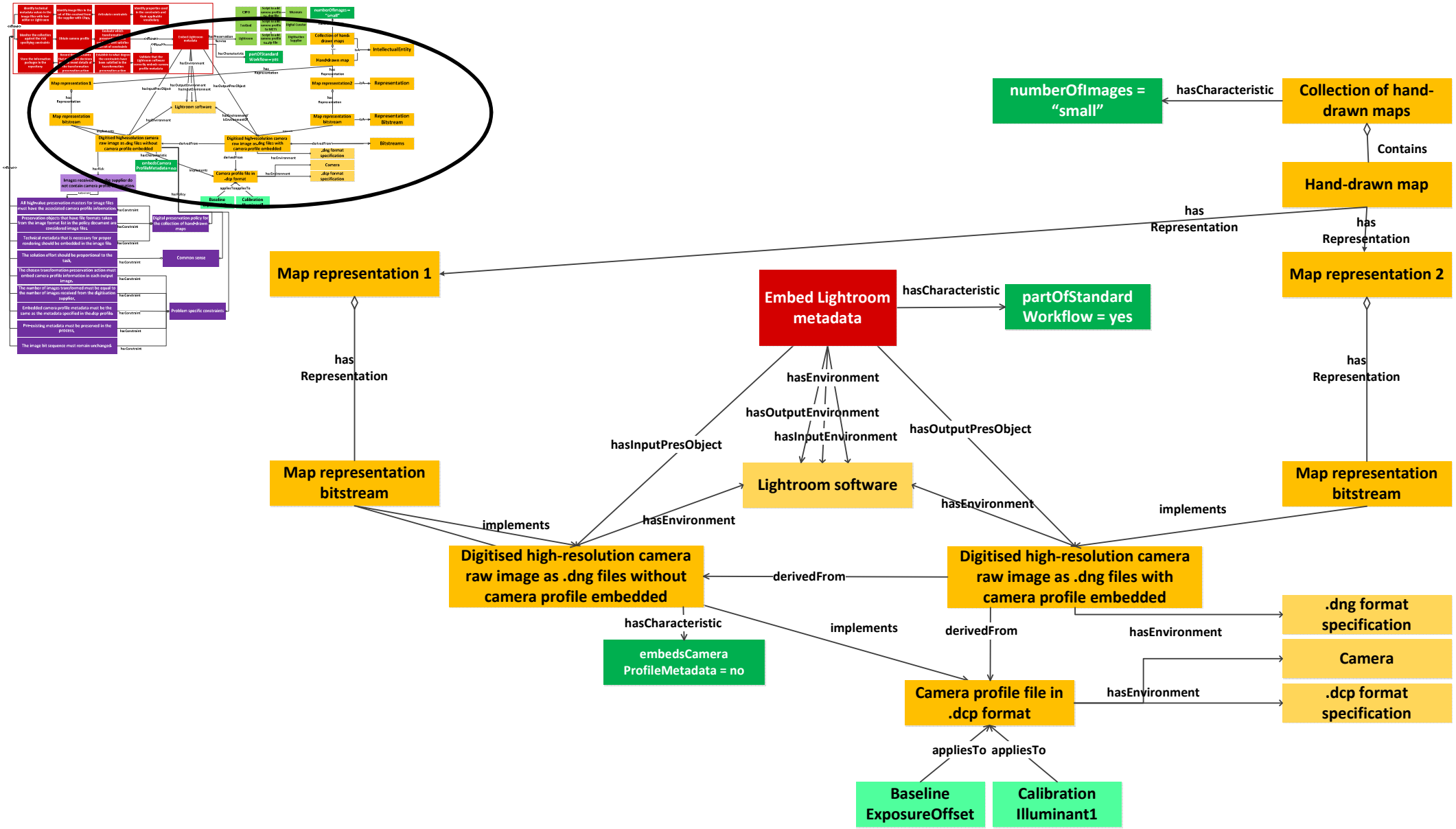



**Figure 52: Vignette 4: *PreservationObjects* and *Environments* involved in the *TransformationPreservationAction* in the example scenario**

# 8 BIBLIOGRAPHY


Abrams, S. L., Morrissey, S., Cramer, T. (2009). "What? So What": The Next-Generation JHOVE2 Architecture for Format-Aware Characterization. International Journal of Digital Curation, 4(3), 123-136. UKOLN. Retrieved from http://ijdc.net/index.php/ijdc/article/download/139/174, accessed on 26 October 2012

Activiti (nd). Activiti. Retrieved from http://www.activiti.org/, accessed on 26 October 2012

Adobe (nd). Digital photography, photo software | Adobe Photoshop Lightroom 4. Retrieved from http://www.adobe.com/uk/products/photoshop-lightroom.html, accessed on 13 April 2013

Ainsworth, A. B., Avram, C., Sheard, J. (2010). The Monash University Museum of Computing: Ten Years On. History of Computing: Learning from the Past. Ed. A. Tatnall. Germany: Springer.

Aitken, B., Helwig, P., Jackson, A., Lindley, A., Nicchiarelli, E., Ross, S. (2008). The Planets Testbed: Science for Digital Preservation. Code4Lib, 1(3). Retrieved from http://journal.code4lib.org/articles/83, accessed on 26 October 2012

AMINET (nd). Aminet. Retrieved from http://aminet.net/, accessed on 23 November 2012

Anderson, D., Delve, J., Konstantelos, L., Lange, A., Bergmeyer, W. (2010). D3.3 Final document analyzing and summarizing metadata standards and issues across Europe. KEEP project deliverable D3.3. Retrieved from http://www.keep-project.eu/ezpub2/index.php?/eng/content/download/19829/99338/file/KEEP_WP3_D3.3_v1.0.pdf, accessed on 16 November 2012

Anderson, D., Delve, J., Pinchbeck, D. (2010). Toward A Workable Emulation-Based Preservation Strategy: Rationale and Technical Metadata. In New Review of Information Networking, Volume 15, Issue 2, pages 110-131. Routledge. ISSN 1361-4576 (Print), 1740-7869 (Online). DOI:10.1080/13614576.2010.530132. Retrieved from http://hnk.ffzg.hr/bibl/InFuture2011/KEEP_project/Further%20Reading/Anderson%20Delve%20and%20Pinchbeck%20emulation%20pre-print.pdf, accessed on 10 August 2012

Anderson, D. (2011). D2.6 A layman's guide to the KEEP legal studies. KEEP Project deliverable. Retrieved from http://www.keep-project.eu/ezpub2/index.php?/eng/content/download/20703/103715/file/D2.6_laymansguidelegalstudies_final.pdf, accessed on 10 August 2012

Anderson D., Delve J., Powell V. (2012). The Changing Face of the History of Computing: The Role of Emulation in Protecting Our Digital Heritage, in Arthur Tatnall (Ed.): Reflections on the History of Computing. Preserving Memories and Sharing Stories. Series: IFIP Advances in Information and Communication Technology, Vol. 387, pp. 362–384. ISBN 978-3-642-33898-4

ANSI/EIA-649A (2011). National Consensus Standard for Configuration Management . TechAmerica. G33- Data & Configuration Management Committee. 01-April-2011.

ANSI/NISO (2006). ANSI/NISO Z39.87. Retrieved from http://www.niso.org/apps/group_public/download.php/6502/Data%20Dictionary%20-%20Technical%20Metadata%20for%20Digital%20Still%20Images.pdf, accessed on 31 August 2012

AQuA (nd). Retrieved from http://wiki.opf-labs.org/display/AQuA/Home, accessed on 10 August 2012

Armadillo Systems (nd). Turning the Pages. Retrieved from http://www.turningthepages.com/, accessed on 9 November 2012

Barateiro, J., Antunes, G., Freitas, F., Borbinha, J. (2010). Designing Digital Preservation Solutions: A Risk Management-Based Approach. The International Journal of Digital Curation. Issue 1, Volume 5. 2010. Retrieved from ijdc.net/index.php/ijdc/article/download/143/205, accessed on 9 November 2012

Barateiro, J., Borbinha, J. (2011). Integrated management of risk information. IEEE. Proceedings of the Federated Conference on Computer Science and Information Systems pp. 801–808, ISBN 978-83-60810-22-4

Beagrie, N., Semple, N., Williams, P., Wright, R. (2008). Digital Preservation Policies Study. Part 1: Final Report October 2008. Prepared by: Charles Beagrie Limited, www.beagrie.com. A study funded by JISC. Copyright HEFCE 2008. Retrieved from http://www.jisc.ac.uk/media/documents/programmes/preservation/jiscpolicy_p1finalreport.pdf, accessed on 16 November 2012

Becker, C., Rauber, A., Heydegger, V., Schnasse, J., Thaller, M. (2008a). A generic XML language for characterising objects to support digital preservation. Proceedings of the 2008 ACM symposium on Applied computing - SAC ’08. Fortaleza; Cear´a; Brazil. (p. 402). New York, New York, USA: ACM Press. doi:10.1145/1363686.1363786

Becker, C., Kulovits, H., Rauber, a., Hofman, H. (2008b). Plato: A Service Oriented Decision Support System for Preservation Planning. Proceedings of the 8th ACMIEEECS Joint Conference on Digital Libraries (pp. 367-370). Pittsburgh; Pennsylvania; USA: ACM Press. doi:10.1145/1378889.1378954

Becker, C., Duretec, K., Petrov, P., Faria, L., Ferreira, M., Ramalho, J.C. (2012). Preservation Watch: What to monitor and how. iPRES 2012: The Ninth International Conference on Preservation of Digital Objects. Retrieved from https://ipres.ischool.utoronto.ca/sites/ipres.ischool.utoronto.ca/files/iPres%202012%20Conference%20Proceedings%20Final.pdf, accessed on 23 November 2012

Bezivin, J Muller, P.-A. (Eds) (1999). The Unified Modeling Language. '98: Beyond the Notation: First International Workshop. Lecture Notes in Computer Science, Vol. 1618. Springer Verlag, 30 July 1999. 443 pages. ISBN-13 9783540662525

Bradley, K., & Woodyard, D. (2000). Preservation Metadata for Digital Collections. Metadata for long term preservation in NEDLIB. Paris: Bibliotheque national de France. Retrieved from http://www.nla.gov.au/openpublish/index.php/nlasp/article/view/1344/1628, accessed on 7 December 2012

Bolton University (nd). Archi. Retrieved from http://archi.cetis.ac.uk/, accessed on 16 November 2012

Bray, T., Paoli, J., Sperberg-McQueen , C.M., Maler, E., Yergeau, F. (Eds) (2008). Extensible Markup Language (XML). 1.0 (Fifth Edition). W3C Recommendation. 26 November 2008. Copyright © 2008 W3C® Retrieved from http://www.w3.org/TR/xml/, accessed on 2 November 2012

British Library (2007). British Library Digital Preservation Strategy (pp. 1-3). Retrieved from http://www.webarchive.org.uk/wayback/archive/20120724144603/http://www.bl.uk/aboutus/stratpolprog/ccare/introduction/digital/digpresstrat.pdf, accessed on 30 November 2012

Brown, A. (2005). Automating preservation: New developments in the PRONOM service, RLG DigiNews 9(2). Retrieved from http://worldcat.org/arcviewer/1/OCC/2007/07/10/0000068902/viewer/file1.html#article, accessed on 9 November 2012

Brown, A. (2008). White Paper: Representation Information Registries. PLANETS report PC/3-D7. Retrieved from http://www.planets-project.eu/docs/reports/Planets_PC3-D7_RepInformationRegistries.pdf

Brown, G., Woods, K. (2011). Born Broken: Fonts and Information Loss in Legacy Digital Documents. International Journal of Digital Curation, 6(1). Retrieved from http://ijdc.net/index.php/ijdc/article/viewFile/159/243, accessed on 23 November 2012

Burtles, J. (2007). Principles and Practice of Business Continuity: Tools and Techniques. Rothstein Associates Inc., ISBN-13: 978-1931332392

Canteiro, S., Barateiro, J. (2011). Risk Assessment in Digital Preservation of e-Science Data and Processes. iPRES 2011: The Eighth International Conference on Preservation of Digital Objects. Retrieved from http://ipres2011.sg/conference-procedings, accessed on 17 August 2012

Caplan, P. (2006), Preservation Metadata. DCC Digital Curation Manual, Version 1.0. Warwick, UK. S.Ross, M.Day (eds). Publisher: HATII, University of Glasgow; University of Edinburgh; UKOLN, University of Bath; Council for the Central Laboratory of the Research Councils. http://hdl.handle.net/1842/3356, ISSN 1747-1524. Retrieved from http://www.dcc.ac.uk/sites/default/files/documents/resource/curation-manual/chapters/preservation-metadata/preservation-metadata.pdf, accessed on 30 November 2012

Caplan, P. (2008). The Preservation of Digital Materials. Library Technology Reports, Vol. 44, No. 2. (February 2008), p. 9

CCSDS (2002). Reference Model for an Open Archival Information System (OAIS). CCSDS 650.0-B-1. Retrieved from http://public.ccsds.org/publications/archive/650x0b1.pdf, accessed on 30 November 2012

CCSDS (2009). Reference Model for an Open Archival Information System (OAIS). Draft Recommendation for Space Data System Standards. CCSDS 650.0-P-1.1. (Pink Book) Issue 1.1, August 2009, Washington, DC. Retrieved from http://public.ccsds.org/sites/cwe/rids/Lists/CCSDS 6500P11/Attachments/650x0p11.pdf,

http://public.ccsds.org/sites/cwe/rids/Lists/CCSDS 6500P11/CCSDSAgency.aspx, , accessed on 30 November 2012

CCSDS (2011). Audit and Certification of Trustworthy Digital Repositories. Recommended Practice. CCSDS 652.0-M-1. Magenta Book.September 2011. Retrieved from http://public.ccsds.org/publications/archive/652x0m1.pdf, accessed on 9 November 2012

CCSDS (2012). Reference Model for an Open Archival Information System (OAIS): version 2. CCSDS 650.0-B-1, Blue Book (the full ISO standard). Retrieved from http://public.ccsds.org/publications/archive/650x0m2.pdf, accessed on 10 August 2012

CEDARS (nd). The CEDARS Project. CURL Exemplars in Digital ARchiveS. Website. UKOLN. Retrieved from http://www.ukoln.ac.uk/services/elib/projects/cedars/, accessed on 24 August 2012

CEDARS Project. (2002). Cedars Guide to Preservation Metadata. Retrieved from http://www.webarchive.org.uk/wayback/archive/20050410120000/http://www.leeds.ac.uk/cedars/guideto/metadata/index.html

Center for Research Libraries (CRL) and Online Computer Library Center Inc. (OCLC) (2007). Trustworthy Repositories Audit & Certification: Criteria and Checklist. Version 1.0. February 2007. Retrieved from http://www.crl.edu/PDF/trac.pdf, accessed on 1 November 2012

Charlesworth, A. (2012). Intellectual Property Rights and Preservation. DPC Technology Watch Report 12-02. Retrieved from http://www.dpconline.org/publications/technology-watch-reports, accessed on 10 August 2012 (login required)

Chaudhri, V., Farquhar, A., Fikes, R., Karp, P., Rice, J. (1998). OKBC: A programmatic foundation for knowledge base interoperability. Proceedings of the 1998 National Conference on Artificial Intelligence . Menlo Park, CA: American Association for Artificial Intelligence . Retrieved from http://www.stanford.edu/class/cs227/Readings/405.pdf, accessed on 30 November 2012

Chen, P. P. (1976). The Entity-Relationship Model: Toward a Unified View of Data. In ACM Transactions on Database Systems (TODS) - Special issue: papers from the international conference on very large data bases. September 22–24, 1975, Framingham, MA.Volume 1 Issue 1, March 1976.Page s 9-36. ACM New York, NY, USA. DOI 10.1145/320434.320440

Chue Hong, N., Crouch, S., Hettrick, S., Parkinson, T., Shreeve, M. (2010). Software Preservation, Benefits framework. Software Sustainability Institute and Curtis+Cartwright Consulting Ltd. Report CC443D006-01 Retrieved from http://www.software.ac.uk/attach/SoftwarePreservationBenefitsFramework.pdf. Accessed on 3 August 2012

Christensen, E., Curbera, F., Meredith, G., Weerawarana, S. (2001). Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001. Copyright© 2001 Ariba, International Business Machines Corporation, Microsoft. Retrieved from www.w3.org/TR/wsdl, accessed on 26 October 2012

Cisco Systems (2008). Cisco Application Profiling Service. Service Data Sheet. Retrieved from http://www.cisco.com/en/US/services/ps6887/ps6892/application_profiling_service_data_sheet.pdf, accessed on 16 November 2012

Clausen, L. (2007). Opening Schroedinger's Library: Semi-automatic QA reduces un-certainty in object-transformation. In Kovács, L., Fuhr, N., Meghini, Carlo. (2007). Research and Advanced Technology for Digital Libraries: 11th European Conference on Digital Libraries (ECDL). 2007, Budapest, Hungary, September 16-21, 2007, Proceedings (p. 585). Springer Verlag, Berlin Heidelberg. doi:10.1007/978-3-540-87599-4

CORDIS (nd). Retrieved from http://cordis.europa.eu/fp7/ict/telearn-digicult/digicult-preservation_en.html, accessed on 17 August 2012

Cornell University Library, ICPSR (nd). Digital preservation management: Implementing short-term strategies for long-term problems, Tutorial. Retrieved from http://api.wayback.archive.org/memento/20071209163415/http://www.icpsr.umich.edu/dpm/dpm-eng/eng_index.html, accessed on 10 August 2012

Dappert, A., Ballaux, B., Mayr, M., van Bussel, S. (2008). Report on policy and strategy models for libraries; archives and data centres. Planets report PP2-D2. Retrieved from http://www.planets-project.eu/docs/reports/Planets_PP2_D2_ReportOnPolicyAndStrategyModelsM24_Ext.pdf, accessed on 23 November 2012

Dappert, A. (2009). Report on the Conceptual Aspects of Preservation, Based on Policy and Strategy Models for Libraries, Archives and Data Centres. PLANETS report PP2-D3. Retrieved from http://www.planets-project.eu/docs/reports/Planets_PP2_D3_ReportOnPolicyAndStrategyModelsM36_Ext.pdf, accessed on 23 November 2012

Dappert, A., Farquhar, A. (2009a). Modelling Organizational Preservation Goals to Guide Digital Preservation. International Journal of Digital Curation, 4(2), 119-134. doi:10.2218/ijdc.v4i2.102.Retrieved from http://www.ijdc.net/index.php/ijdc/article/download/123/126, accessed on 23 November 2012

Dappert, A., Farquhar, A. (2009b). Significance Is in the Eye of the Stakeholder. (M. Agosti & et alii, Eds.) ECDL'09 Proceedings of the 13th European conference on Research and advanced technology for digital libraries, September/October 2009, LNCS 5714, 297-308. Berlin, Heidelberg: ©Springer Verlag. Retrieved from http://www.planets-project.eu/docs/papers/Dappert_SignificantCharacteristics_ECDL2009.pdf or http://www.bl.uk/aboutus/stratpolprog/ccare/pubs/2009/ipres2009-Dappert%20and%20Farquhar.pdf, accessed on 8 May 2012

Dappert, A. (2010). Deal With Conflict, Capture the Relationship: The Case of Digital Object Properties. iPRES 2010: The Seventh International Conference on Preservation of Digital Objects Retrieved from http://www.ifs.tuwien.ac.at/dp/ipres2010/papers/dappert-05.pdf, accessed on 23 November 2012

Dappert, A., Enders, M. (2010). Digital Preservation Metadata Standards, NISO Information Standards Quarterly. 22(2), June 2010. Retrieved from http://www.loc.gov/standards/premis/FE_Dappert_Enders_MetadataStds_isqv22no2.pdf accessed on 10 August 2012

Dappert, A., Farquhar, A. (2011). Implementing Metadata that Guide Digital Preservation Services. International Journal of Digital Curation, 6(1). Retrieved from http://ijdc.net/index.php/ijdc/article/view/176, accessed on 30 November 2012

Dappert, A. (2011). Risk Assessment of Digital Holdings. Retrieved from http://www.digitalpreservationsummit.de/presentations/dappert.pdf, accessed on 17 August 2012

Dappert, A., Jackson, A., Kimura, A. (2011). Developing a Robust Migration Workflow for Preserving and Curating Hand-held Media, iPres 2011, The Eighth International Conference on Preservation of Digital Objects. Retrieved from http://ipres2011.sg/conference-procedings#, accessed on 16 November 2012

Dappert, A., Peyrard, S., Delve, J., Chou, C. (2012). Describing Digital Object Environments in PREMIS. iPRES 2012: The Ninth International Conference on Preservation of Digital Objects. Retrieved from https://ipres.ischool.utoronto.ca/sites/ipres.ischool.utoronto.ca/files/iPres%202012%20Conference%20Proceedings%20Final.pdf, accessed on 23 November 2012 2012

Dappert, A. (2012). Proposed Data Model Changes for PREMIS 3.0. (PREMIS Implementation Fair). 2 October 2012. Presentation slides. Retrieved from http://timbusproject.net/component/docman/doc_download/73-proposed-data-model-changes-for-premis-30, accessed on 14 December 2012

DC (nd). Dublin Core (Dublin Core Metadata Initiative – DCMI). Retrieved from www.dublincore.org/, accessed on 31 August 2012

DCC (nd). Home > Resources for digital curators > Digital curation resources from outside the DCC > Tools & Services. Retrieved from http://www.dcc.ac.uk/resources/external/tools-services, accessed on 1 August 2012

Debian (nd). Debian Policy Manual, chapter 7 - Declaring relationships between packages, Debian Policy Manual, version 3.9.3.1, 2012-03-04. Retrieved from http://www.debian.org/doc/debian-policy/ch-relationships.html, accesed on 23 November 2012

Delve, J. (2011). Environment Metadata for Emulation. KEEP Newsletter, Third issue, February 2011. Retrieved from http://www.keep-project.eu/downloads/Newsletter_KEEP_Feb2011.pdf, accessed on 24 August 2012

Delve, J., Anderson, D. (2012). TOTEM. Trusworthy Online Technical Environment Metadata. The Trusworthy Online Technical Environment Metadata Database – TOTEM. Verlag Dr. Kovač, Hamburg, 2012. 479 pages, ISSN 1866-2129, ISBN 978-3-8300-6418-3

Demant, D. (2010). Why the Real Thing is Essential for Telling our Stories. History of Computing: Learning from the Past. Ed. A. Tatnall. Germany: Springer.

Deutsche National Bibliothek (nd). LMER, Langzeitarchivierungsmetadaten für elektronische Ressourcen. (urn:nbn:de:1111-2005051906). Retrieved from http://www.dnb.de/DE/Standardisierung/LMER/lmer_node.html, accessed on 31 August 2012

Donnelly, M. (2010). DCC Case Study – JSTOR/Harvard Object Validation Environment (JHOVE). HATII, University of Glasgow; University of Edinburgh; UKOLN, University of Bath; Council for the Central Laboratory of the Research Councils. Retrieved from http://hdl.handle.net/1842/3335, accessed on 30 November 2012

Dumbill, E. (nd). DOAP- Description of a Project. Retrieved from https://github.com/edumbill/doap/wiki, accessed on 23 November 2012

Duval, E. (2001). Metadata Standards, What, Who & Why, Journal of Universal Computer Science, 7(7), 591-601, Springer Verlag

EAD (2002). EAD: Encoded Archival Description Version 2002 Official Site. Retrieved from http://www.loc.gov/ead/, accessed on 31 August 2012

Edelstein, O., Factor, M.,King, R., Risse,T., Salant, E., Taylor, P. (2011). Evolving Domains, Problems and Solutions for Long Term Digital Preservation. iPRES 2011: The Eighth International Conference on Preservation of Digital Objects. Retrieved from http://ipres2011.sg/conference-procedings, accessed on 17 August 2012

Electronic Resource Preservation and Access Network (2003). ERPA Guidance. Digital Preservation Policy Tool. September 2003. Retrieved from http://www.erpanet.org/guidance/docs/ERPANETPolicyTool.pdf, accessed on 1 November 2012

Farquhar, A. (2007). Sustainable models for digital preservation. Sustainability Models for Digital Preservation, Brussels, 29-30 Nov, 2007. Presentation slides. Retrieved from http://research.microsoft.com/en-us/um/cambridge/events/rpp/workshops/planets_sustainabilityworkshop/Planets_Sustainability_Intro-AdamFarquhar.ppt, accessed on 17 August 2012

Farquhar, A., Hockx-Yu, H. (2007). Planets: Integrated services for digital preservation. International Journal of Digital Curation, 2(2), 88-99. UKOLN. Retrieved from http://www.ijdc.net/index.php/ijdc/article/viewFile/45/31, accessed on 23 November 2012

Fadeyev, V., Haber, C., et al. (2003). Reconstruction of mechanically recorded sound by image processing. Journal of the Audio Engineering Society 51 (December): 172. Bibcode 2001ASAJ.115.2494F. Retrieved from http://www-cdf.lbl.gov/~av/JAES-paper-LBNL.pdf, accessed on 10 August 2012

Fayzullin, M., (1997-2000). How To Write a Computer Emulator. Retrieved from http://fms.komkon.org/EMUL8/HOWTO.html, accessed on 10 August 2012

Florida Digital Archive (2006). Florida Digital Archive (FDA). Policy and Procedures Guide. Version 3.0, May, 201. Retrieved from http://wayback.archive-it.org/316/20060602021421/http://www.fcla.edu/digitalArchive/pdfs/DigitalArchivePolicyGuide.pdf, accessed on 1 November 2012

Florida Digital Archive (2011). Florida Digital Archive (FDA). Policy and Procedures Guide. Version 3.0, May, 201. Retrieved from http://fclaweb.fcla.edu/uploads/FDAPolicyGuideversion3.0.pdf, accessed on 1 November 2012

FRBR (1998). Functional Constraints for Bibliographic Records (Vol. 19). IFLA Study Group on the Functional Requirements for Bibliographic Records. München: UBCIM publications, K.G. Saur. Retrieved from http://archive.ifla.org/VII/s13/frbr/frbr.pdf, accessedon 23 November 2012

Garrett, J., D. Waters, H. Gladney, P. Andre, H. Besser, N. Elkington, H. Gladney, M. Hedstrom, P. Hirtle, K. Hunter, R. Kelly, D. Kresh, M. Lesk, M. Levering, W. Lougee, C. Lynch, C. Mandel, S. Mooney, A. Okerson, J. Neal, S. Rosenblatt, and S. Weibe (1996). Preserving Digital Information. Report of the Task Force on Archiving of Digital Information, commissioned by The Commission on Preservation and Access and The Research Libraries Group. May 1, 1996. Retrieved from http://www.oclc.org/resources/research/activities/digpresstudy/final-report.pdf, accessed on 2 November 2012

Guttenbrunner, M., Wieners, J., Rauber, A., Thaller, M. (2010). Same Same But Different – Comparing Rendering Environments. M. Ioannides (Ed.): EuroMed 2010, LNCS 6436, pp. 140–152, 2010. Springer-Verlag Berlin Heidelberg 2010

Guttenbrunner, M., Rauber, A. (2012). A Measurement Framework for Evaluating Emulators for Digital Preservation. ACM Transactions on Information Systems (TOIS) TOIS. Volume 30 Issue 2, May 2012, Article No. 14, ACM, New York, NY, USA

Hampshire Record Office (2005). Hantsweb. Digital Preservation Policy. Agreed by HRO Management Team, August 2005. Revised August 2010. Retrieved from http://www3.hants.gov.uk/archives/hro-policies/hro-digital-preservation-policy.htm, accessed on 1 November 2012

Heydegger, V. (2009). Just One Bit in a Million: On the Effects of Data Corruption in Files. In M. Agosti (Ed.), European Conference on Digital Libraries (ECDL), Research and Advanced Technology for Digital … (pp. 315-326). Berlin Heidelberg: Springer-Verlag LNCS 5714.

Hillah, M. L,Kindler, E., Kordon, F., Petrucci, L., Trèves, N. (2009). A primer on the Petri Net Markup Language and ISO/IEC 15909-2. Petri Net Newsletter 76:9--28, October 2009. Retrieved from http://www.pnml.org/papers/pnnl76.pdf, accessed on 26 October 2012

Hitchcock, S., Tarrant, D. Carr, L., Kulovits, H., Rauber, A. (2010). Connecting preservation planning and Plato with digital repository interfaces. iPRES 2010: The Seventh International Conference on Preservation of Digital Objects. Retrieved from http://eprints.soton.ac.uk/271289/, accessed on 9 November 2012

HM Government (2008). Managing information risk: A guide for accounting officers, board members and senior information risk owners. Retrieved from http://www.nationalarchives.gov.uk/services/publications/information-risk.pdf, accessed on 17 August 2012

Hockx-Yu, H., Knight, G. (2008). What to Preserve? Significant Properties of Digital Objects. Report on the JISC/BL/DPC Workshop of April 7, 2008, British Library Conference Centre. International Journal of Digital Curation, 3(1), 141-153. UKOLN. Retrieved from http://www.ijdc.net/index.php/ijdc/article/view/70/49 or http://www.ijdc.net/index.php/ijdc/article/download/70/49, accessed on 23 November 2012

Hodgson, R., Keller, P. (2011). QUDT - Quantities, Units, Dimensions and Data Types in OWL and XML. TopQuadrant and NASA. September 11, 2011. Retrieved from www.qudt.org/, accessed on 9 November 2012

Hoffman, J. (2012). Utility Spotlight. RichCopy. Free Utility: RichCopy, an Advanced Alternative to Robocopy. Microsoft TechNet Magazine. Issue April, 2009. Retrieved from http://technet.microsoft.com/en-us/magazine/2009.04.utilityspotlight.aspx, accessed on 16 November 2012

Huth, K. (2004). Probleme and Lösungsansätze zur Archivierung von Computerprogrammen—am Beispiel der Software des ATARI VCS 2600 und des C64. Berlin: Humboldt-Universität zu Berlin, 2004.

IBM (nd). Rational System Architect. Retrieved from http://www-01.ibm.com/software/awdtools/systemarchitect/, accessed on 16 November 2012

IBM (2011). IBM Tivoli Endpoint Manager for Software Use Analysis. Rapid, granular inventory insights and always-on asset management enhance license compliance. Data sheet TID14086-USEN-00. IBM. October 2011. Retrieved from http://public.dhe.ibm.com/common/ssi/ecm/en/tid14086usen/TID14086USEN.PDF, accessed on 16 November 2012

IIPC Preservation Working Group (nd). Retrieved from http://netpreserve.org/about/pwg.php, accessed on 26 October 2012

ImageMagick (nd). ImageMagick: Convert, Edit, Or Compose Bitmap Images. Retrieved from http://www.imagemagick.org/script/index.php, accessed on 2 November 2012

InterPARES (nd). The InterPARES 2 Project. Retrieved from http://www.interpares.org/ip2/display_file.cfm?doc=ip2_glossary.pdf&CFID=243105&CFTOKEN=70677126, accessed on 27 September 2007

ISO (2005). ISO/IEC 27001:2005 Information technology – Security techniques – Information security management systems – Requirements. Can be purchased from http://www.iso.org/iso/catalogue_detail?csnumber=42103, accessed on 17 August 2012

ISO (2009). ISO 31000. Risk management -- Principles and guidelines. International Organization for Standardization. Can be purchased from http://www.iso.org/iso/catalogue_detail.htm?csnumber=43170, accessed on 17 August 2012

ISO 13008:2012 (2012). Information and documentation -- Digital records conversion and migration process. Can be purchased from http://www.iso.org/iso/catalogue_detail.htm?csnumber=52326, accessed on 7 December 2012

ISO/IEC (2005). ISO/IEC 21000-2:2005. MPEG-21 DIDL. Information technology -- Multimedia framework (MPEG-21) -- Part 2: Digital Item Declaration. Retrieved from www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=35366, accessed on 31 August 2012

IT Governance Institute (2007). CobiT 4.1. Framework – Control Objectives – Management Guidelinces – Maturity Models. Retrieved from http://www.trainning.com.br/download/COBIT_41.pdf, accessed on 23 November 2012

ITU Radiocommunication Assembly (2002). RECOMMENDATION ITU-R BT.500-11. Methodology for the subjective assessment of the quality of television pictures (Question ITU-R 211/11). (1974-1978-1982-1986-1990-1992-1994-1995-1998-1998-2000-2002). Retrieved from http://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.500-11-200206-S!!PDF-E.pdf, accessed on 23 November 2012

JISC (2006). Digital Preservation, Continued access to authentic digital assets. Briefing paper. November 2006. Retrieved from http://www.jisc.ac.uk/media/documents/publications/digitalpreservationbp.pdf, accessed on 1 November 2012

Jones, M., Beagrie, N. (2001) Preservation Management of Digital Materials: A Handbook. The Council for Museums, Archives and Libraries. ISBN: 0-7123-0886-5.

Jones, M., Beagrie, N. (2008). Preservation Management of Digital Materials: A Handbook. (The Digital Preservation Coalition, Ed.). Retrieved from http://www.dpconline.org/component/docman/doc_download/299-digital-preservation-handbook-digital-preservation-handbook?q=handbook, accessed on 23 November 2012

JSTOR, Harvard University Library (nd). JHOVE - JSTOR/Harvard Object Validation Environment. Retrieved from http://hul.harvard.edu/jhove/, accessed 8 May 2012

Kadobayashi, Y. (2010). Toward Measurement and Analysis of Virtualized Infrastructure: Scaffolding from an Ontological Perspective. Retrieved from http://www.caida.org/workshops/wide-casfi/1004/slides/wide-casfi1004_ykadobayashi.pdf, accessed on 8 May 2012

KEEP (nd). Keeping Emulation Environments Portable. Retrieved from http://www.keep-project.eu/ezpub2/index.php, accessed on 8 May 2012

Kirschenbaum, M.G. Ovenden, R., Redwine, G. (2010). Digital Forensics and Born-Digital Content in Cultural Heritage Collections. Council on Library and Information Resources Washington, D.C., ISBN 978-1-932326-37-6. Retrieved from http://www.clir.org/pubs/reports/pub149/reports/pub149/pub149.pdf, accessed on 10 August 2012

Knight, G. (2008). Framework for the definition of significant properties. 5 February 2008. Retrieved from http://www.significantproperties.org.uk/wp33-propertiesreport-v1.pdf, accessed 16 November 2012

Knight, G. (2009). InSPECT Framework Report. 13 October 2009. Retrieved from http://www.significantproperties.org.uk/inspect-framework.pdf, accessed on 16 November 2012

Knight, G., Pennock, M. (2009). Data Without Meaning: Establishing the Significant Properties of Digital Research. International Journal of Digital Curation, 4(1). doi:10.2218/ijdc.v4i1.86; Retrieved from http://www.ijdc.net/index.php/ijdc/article/download/110/87, accessed on 23 November 2012

Kuchera, B. (2011). Accuracy takes power: one man's 3GHz quest to build a perfect SNES emulator. How can it take 3GHz to emulate a Super Nintendo? The man behind a major SNES. Ars Technica, 9 August 2011. Retrieved from http://arstechnica.com/gaming/2011/08/accuracy-takes-power-one-mans-3ghz-quest-to-build-a-perfect-snes-emulator/, accessed on 10 August 2012

Kulovits, H., Rauber, A., Kugler, A., Brantl, M., Beinert, T., Schoger, A. (2009). From TIFF to JPEG 2000? Preservation Planning at the Bavarian State Library Using a Collection of Digitized 16th Century Printings. D-Lib Magazine. November/December 2009. Volume 15, Number 11/12. ISSN 1082-9873. Retrieved from http://www.dlib.org/dlib/november09/kulovits/11kulovits.html, accessed on 3 August 2012

Lavoie, B., Gartner, R. (2005). Preservation Metadata. DPC Technology Watch Report 05-01. OCLC Online Computer Library Center Inc., Oxford University Library Services and Digital Preservation Coalition. Retrieved from http://www.dpconline.org/docs/reports/dpctw05-01.pdf, accessed on 23 November 2012

Lee, C., Chassanoff A., Woods, K., Kirschenbaum, M., Olsen, P. (2012). BitCurator: Tools and Techniques for Digital Forensics in Collecting Institutions. DLIB Magazine, May/June 2012 – Volume 18 Issue 5/6

Library of Congress (nd-a). Sustainability of Digital Formats. Planning for Library of Congress Collections. Retrieved from www.digitalpreservation.gov/formats/, accessed on 10 August 2012

Library of Congress (nd-b). Authorities and Vocabularies. Retrieved from http://id.loc.gov/, accessed on 26 October 2012

LOCKSS (nd). Retrieved from http://www.lockss.org/. Accessed on 3 August 2012

Lorie, R. A. (2001). Long Term Preservation of Digital Information. Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '01). Roanoke, Virginia, USA. pp. 346–352. Retrieved from http://old.hki.uni-koeln.de/teach/ss06/DL/material/p346-lorie.pdf, accessed on 23 November 2012

Ludäscher, B., Altintas I., Berkley C., Higgins D., Jaeger-Frank E., Jones M., Lee E., Tao J., Zhao Y. (2006). Scientific Workflow Management and the Kepler System. Special Issue: Workflow in Grid Systems. Concurrency and Computation: Practice & Experience 18(10): 1039-1065.

Lupovici, C., Masanès, J. (2000). Metadata for the Long Term Preservation of Electronic Publications. The Hague: Koninklije Bibliotheek. NEDLIB Report Series 2. Networked European Deposit Library. doi:10.1.1.170.3781. Retrieved from http://www.kb.nl/sites/default/files/docs/NEDLIBmetadata.pdf, accessed on 23 November 2012

MARC (nd). MARC Standards. Retrieved from http://www.loc.gov/marc/, accessed on 31 August 2012

MARCXML (nd). MARC 21 XML Schema. Retrieved from www.loc.gov/standards/marcxml/, accessed on 31 August 2012

MathArc (nd). MathArc — Ensuring Access to Mathematics Over Time. Retrieved from http://www.library.cornell.edu/dlit/MathArc/web/StoryFrameset.html, accessed on 28 March 2013

Matthews, B., McIlwrath, B., Giaretta, D., Conway, E. (2008). The Significant Properties of Software: A Study. STFC, December 2008. Retrieved from http://www.jisc.ac.uk/media/documents/programmes/preservation/spsoftware_report_redacted.pdf, accessed on 8 May 2012

McDonough, J., et al. (2010). Preserving Virtual Worlds. Retrieved from https://www.ideals.illinois.edu/bitstream/handle/2142/17097/PVW.FinalReport.pdf?sequence=2, accessed on 8 May 2012

McGuinness, D. L., van Harmelen, F. (2004). OWL Web Ontology Language, Overview, W3C Recommendation 10 February 2004. Retrieved from http://www.w3.org/TR/owl-features/, accessed on 26 October 2012

McHugh, A., Innocenti, P., Ross, S. (2008). Assessing risks to digital cultural heritage with DRAMBORA. International Documentation Committee of the International Council of Museums (CIDOC) 2008, Athens, Greece, 15–18 September 2008.

Mediapedia (nd). mediapedia.com. Retrieved from http://www.mediapedia.com/, accessed on 26 October 2012

METS (nd). Metadata Encoding and Transmission Standard (METS) Official Web Site. Retrieved from http://www.loc.gov/standards/mets/, Accessed on 31 August 2012

Miles, R., Hamilton, K. (2006). Learning UML 2.0. O'Reilly Media; 1 edition (2 May 2006). 290 pages. ISBN-13: 978-0596009823

MIX (nd). Metadata for Images in XML Standard (MIX). Retrieved from www.loc.gov/standards/mix/, accessed on 31 August 2012

MobyGames (nd). Retrieved from http://www.mobygames.com/, accessed on 8 May 2012

MODS (nd). Metadata Object Description Schema: MODS (Library of Congress). Retrieved from www.loc.gov/standards/mods/, accessed on 31 August 2012

National Archives of Australia (2005). Digital Preservation Policy, Preserving Archival Digital Records Transferred from Commonwealth Agencies. July 2009. Version 1,2, July 2011. Retrieved from http://wayback.archive-it.org/252/20051028001610/http://www.nla.gov.au/policy/digpres.html, accessed on 1 November 2011

National Archives of Australia (2011). Digital Preservation Policy, Preserving Archival Digital Records Transferred from Commonwealth Agencies. July 2009. Version 1,2, July 2011. Retrieved from http://www.naa.gov.au/about-us/organisation/accountability/operations-and-preservation/digital-preservation-policy.aspx, accessed on 1 November 2011

NCBI & NLM (2012). Journal Publishing Tag Library. NISO JATS Version 1.0. August 2012. National Center for Biotechnology Information (NCBI) and National Library of Medicine (NLM). Retrieved

from http://jats.nlm.nih.gov/publishing/tag-library/1.0/index.html, accessed on 1 November 2011
Retrieved from http://jats.niso.org/,accessed on 23 November 2012

NISO (2004). Understanding Metadata. Available on the NISO website (www.niso.org) and in hardcopy from NISO Press. National Information Standards Organization, 4733 Bethesda Avenue, Suite 300, Bethesda, MD 20814 USA. Copyright © 2004 National Information Standards Organization. ISBN: 1-880124-62-9

NISO (2012). National Information Standards Organization. JATS: Journal Article Tag Suite, ANSI/NISO Z39.96-2012. Retrieved from www.niso.org/workrooms/journalmarkup, accessed on 31 August 2012

NLM (nd). National Center for Biotechnology Information (NCBI) of the National Library of Medicine (NLM). Archiving and Interchange Tag Set. Retrieved from http://dtd.nlm.nih.gov/, accessed on 31 August 2012

Novak, A. (2006). Fixity Checks: Checksums, Message Digests and Digital Signatures. ILTS Digital Preservation Committee. Retrieved from http://www.library.yale.edu/iac/DPC/AN_DPC_FixityChecksFinal11.pdf, accessed on 3 August 2012

NSRL (nd). National Software Reference Library (NSRL). Data Formats of the NSRL Reference Data Set (RDS) Distribution. Retrieved from http://www.nsrl.nist.gov/Documents/Data-Formats-of-the-NSRL-Reference-Data-Set-16.pdf, accessed on 2 November 2012

Object Management Group (2011a). Business Process Model and Notation (BPMN). Version 2.0. Release date: January 2011. Retrieved from http://www.omg.org/spec/BPMN/2.0/PDF, accessed on 26 October 2012

Object Management Group (2011b). OMG Unified Modeling Language. TM (OMG UML), Infrastructure. Version 2.4.1. OMG Document Number: formal/2011-08-05. Standard document URL: http://www.omg.org/spec/UML/2.4.1/Infrastructure. Copyright © 1997-2011 Object Management Group, accessed on 30 November 2012

OBO Foundry Initiative (nd). unit-ontology. Ontology of Units of Measurement. Retrieved from http://code.google.com/p/unit-ontology/, accessed on 9 November 2012

OCLC/RLG Working Group on Preservation Metadata (2002). Preservation Metadata and the OAIS Information Model, A Metadata Framework to Support the Preservation of Digital Objects. Online Computer Library Center/Research Libraries Group [OCLC/RLG]. Retrieved from http://www.oclc.org/research/activities/past/orprojects/pmwg/pm_framework.pdf, http://www.oclc.org/research/projects/pmwg/pm_framework.pdf, accessed on 31 August 2012

Open Group (2012). ArchiMate® 2.0 Specification. Jan 2012. Retrieved from https://www2.opengroup.org/josso/signon/login.do?josso_back_to=https://www2.opengroup.org/ogsys/josso_security_check&josso_partnerapp_host=www2.opengroup.org&josso_partnerapp_ctx=/ogsys, accessed on 9 November 2012

PADI (nd). Preserving Access to Digital Information (PADI), National Library of Australia. Preservation metadata for digital collections : exposure draft. http://pandora.nla.gov.au/tep/25498 was selected for preservation by the National Library of Australia. Archived on 25 Jun

PARS (2007). The Working Group; Preservation and Reformatting Section. Definitions of Digital Preservation. American Library Association (ALA), Washington, D.C. Retrieved from http://www.ala.org/ala/mgrps/divs/alcts/resources/preserv/defdigpres0408.pdf, accessed on 10 August 2012

Planets - XCL project (nd).The XCL Ontology | XCL - eXtensible Characterization Language. Retrieved from http://planetarium.hki.uni-koeln.de/planets_cms/xcl-ontology, accessed on 9 November 2012

Planets (n.d.). PLANETS: Home. Project website. Retrieved from http://www.planets-project.eu/, accessed on 29 April 29 2011

POCOS (nd). Preservation of Complex Objects Symposia. Retrieved from http://www.pocos.org/, accessed on 26 October 2012

Poole, J. D. (2001).Model-Driven Architecture: Vision, Standards and Emerging Technologies. Position Paper Submitted to ECOOP 2001. Workshop on Metamodeling and Adaptive Object Models. April 2001. Retrieved from http://www.omg.org/mda/mda_files/Model-Driven_Architecture.pdf, accessed on 2 November 2012

PREMIS (nd). PREMIS. Retrieved from http://www.loc.gov/standards/premis/, accessed on 26 October 2012

PREMIS (2005). PREMIS Data Dictionary for Preservation Metadata (Version 1). Retrieved from http://www.oclc.org/research/activities/past/orprojects/pmwg/premis-final.pdf, accessed on 26 October 2012

PREMIS (2008) PREMIS Data Dictionary for Preservation Metadata, Version 2. March 2008. Retrieved from http://www.loc.gov/standards/premis/v2/premis-2-0.pdf, accessed on 26 October 2012

PREMIS (2011). PREMIS Data Dictionary for Preservation Metadata Version 2.1. Retrieved from http://www.loc.gov/standards/premis/v2/premis-2-1.pdf, accessed on 26 October 2012

PREMIS (2012). PREMIS Data Dictionary for Preservation Metadata, Version 2.2. Retrieved from http://www.loc.gov/standards/premis/v2/premis-2-2.pdf, accessed on 26 October 2012

Prom, C. (2010). PLATO (Digital Preservation Planning) Software Review. On April 25, 2010, in Practical E-Records, software and tools for archivists. Retrieved from http://e-records.chrisprom.com/plato-digital-preservation-planning-software-review/, accessed on 1 November 2012

Reckwerdt, B. (nd). Quantitative Picture Quality Assessment Tools. Retrieved from www.videoclarity.com/CSRTMLongDurationTesting.html, accessed on 9 November 2012

Rosenthal, D.S.H. (2010). Bit Preservation: A Solved Problem?. The International Journal of Digital Curation. Issue 1, Volume 5, 2010. UKOLN at the University of Bath, Digital Curation Centre. ISSN: 1746-8256; Retrieved from

http://www.bl.uk/ipres2008/presentations_day2/43_Rosenthal.pdf, accessed on 23 November 2012

Rothenberg, J. (1998). “Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation.” Council on Library and Information Resources, 1998. Retrieved from http://www.clir.org/pubs/reports/rothenberg/contents.html, accessed on 30 November 2012

Rothenberg, J. (2000). Preserving Authentic Digital Information . In Cullen, C., Hirtle, P., Lynch, C., Rothenberg, J. (2000). Authenticity in a Digital Environment (76 pages). Washington, DC: Council on Library and Information Resources. Retrieved from www.clir.org/pubs/reports/pub92/pub92.pdf. Retrieved from http://www.clir.org/pubs/reports/pub92/rothenberg.html, accessed on 16 November 2012

Rusbridge, C. (2006). Excuse Me... Some Digital Preservation Fallacies? Ariadne, February 2(46). UKOLN. ISSN 1361-3200. Retrieved from http://www.ariadne.ac.uk/issue46/rusbridge/, accessed on 23 November 2012

SCAPE (nd). Website. Retrieved from http://www.scape-project.eu/, accessed on 24 August 2012

Searle, S., Thompson, D. (2003). Preservation Metadata. Pragmatic First Steps at the National Library of New Zealand. D-Lib Magazine, April 2003, Volume 9 Number 4, ISSN 1082-9873. Retrieved from http://www.dlib.org/dlib/april03/thompson/04thompson.html, accessed on 31 August 2012

Second Life (nd). Virtual Worlds, Avatars, free 3D chat, online meetings - Second Life Official Site. Retrieved from http://secondlife.com/whatis/?lang=en-US, accessed on 31 August 2012

Sierman, B. (2009). Report on the Planets Functional Model. Retrieved from http://www.planets-project.eu/docs/reports/Planets_PP7-D3-4_ReportOnThePlanetsFunctionalModel.pdf

SNIA (2012). Cloud Data Management Interface (CDMI). Retrieved from http://www.snia.org/cdmi, accessed on 26 October 2012

Software Sustainability Institute, Curtis+Cartwright (nd). Preserving software resources. Retrieved from http://software.ac.uk/resources/preserving-software-resources, accessed on 8 May 2012

Solinet (2005). Contents of a Digital Preservation Policy[digpolicy]. http://www.solinet.net/preservation/preservation_templ.cfm?doc_id=3678, Retrieved From http://api.wayback.archive.org/memento/20061013054210/http://www.solinet.net/preservation/preservation_templ.cfm?doc_id=3678, accessed on 30 November 2012

Sparx Systems (nd). Enterprise Architect. Visual Modeling Platform. Retrieved from http://www.sparxsystems.com/products/ea/index.html, accessed on 16 November 2012

Stawowczyk Long, A., Pearson, D. (2009), I Say Emulate; He Says Migrate – Are Emulation or Migration Feasible Preservation Strategies? Paper presented at the IIPC (International Internet Preservation Consortium) Open Day, 2009. http://www.nla.gov.au/openpublish/index.php/nlasp/article/download/1475/1803, accessed on 23 November 2012

StratML (nd). Retrieved from http://www.xml.gov/stratml/index.htm, accessed on 31 August 2012

StratML (2006). Strategic Markup Language (StratML). Retrieved from http://www.xml.gov/presentations/gpo/stratml20060118.ppt, accessed on 31 August 2012

SWO (nd). The Software Ontology. Retrieved from http://theswo.sourceforge.net/ (2011) and http://sourceforge.net/projects/theswo/files/ (5 October 2012), accessed on 26 October 2012

SWOP (nd). SWOP: The Software Ontology Project. Retrieved from http://www.jisc.ac.uk/whatwedo/programmes/inf11/digpres/swop.aspx, accessed on 26 October 2012

Taverna (nd). Taverna. Retrieved from http://www.taverna.org.uk/, accessed on 26 October 2012

TEI (nd). TEI: Text Encoding Initiative Retrieved from www.tei-c.org/, accessed on 31 August 2012

Tessella (nd). Tessella - Digital Preservation - Safety Deposit Box. Retrieved from http://www.digital-preservation.com/solution/safety-deposit-box/, accessed on 1 November 2012. Details of the underlying conceptual model were made available privately.

textMD (nd). textMD Technical Metadata for Text. Official Website. Retrieved from www.loc.gov/standards/textMD/, accessed on 31 August 2012

Thaller, M., Heydegger, V., Schnasse, J., Beyl, S., Chudobkaite, E. (2008). Significant characteristics to abstract content: Long term preservation of information. In B Christensen-Dalsgaard, D Castelli, B. Jurik, & J Lippincott (Eds.), Lecture Notes in Computer Science: Vol 5173. Research and Advanced Technology for Digital Libraries (pp. 41-49). Berlin, Heidelberg: Springer Verlag.

Thaller, M. (2009). The eXtensible Characterisation Languages – XCL. Verlag Dr. Kovač, Hamburg, 2009. 479 pages, ISBN 978-3-8300-4766-7

TIMBUS (nd). TIMBUS Project. Retrieved from http://timbusproject.net, accessed on 24 August 2012

TNA (nd). PRONOM. The National Archives. Retrieved from www.nationalarchives.gov.uk/PRONOM/Default.aspx, accessed on 31 August 2012

TNA (2011). Risk Assessment Handbook. The National Archives. Crown copyright 2011. Retrieved from http://www.nationalarchives.gov.uk/documents/information-management/Risk-Assessment-Handbook.pdf, accessed on 9 November 2012

TOSEC (nd). The Old School Emulation Centre. What is TOSEC. Retrieved from http://www.tosecdev.org, accessed on 26 October 2012

TOTEM (nd). Welcome to TOTEM - the Trust-worthy Online Technical Environment Metadata Database. Retrieved from http://keep-totem.co.uk, accessed on 8 May 2012

Trezentos, P., Romão, M., Teixeira, R., Palma,A., Schmidtke, H., Alexander Neumann, M. A., Antunes, G., Proença, D., Caetano, A., Nolan, M., Draws, D., Heinrich, G., Mayer, R., Redlich, D. (2012). D4.2: Dependency Models Iter. 1: Definition of a formalism to express dependencies and relations between technological, business and organizational components and processes. TIMBUS project deliverable. Retrieved from http://timbusproject.net/component/docman/doc_download/44-d42m12dependencymodelsiter1pdf-, accessed on 16 November 2012

TU Wien (nd-a). Planets Preservation Planning Tool: Plato 3.0 User Manual, V1.0, June 2010. Retrieved from http://www.ifs.tuwien.ac.at/dp/plato/docs/Plato_3_UserManual.pdf, accessed on 17 August 2012

TU Wien (nd-b). C3PO. Clever, Crafty Content Profiling of Objects. Information Management and Preservation. Retrieved from http://ifs.tuwien.ac.at/imp/c3po, accessed on 13 April 2013

UDFR (nd-a). Unified Digital Format Registry. Retrieved from http://www.udfr.org, accessed on 8 May 2012

UDFR (nd-b). Unified Digital Format Registry UDFR ontology. Retrieved from http://udfr.org/onto/onto.rdf, accessed on 8 May 2012

UK Data Archive (UKDA) (2005). UK Data Archive Preservation Policy. Retrieved from http://api.wayback.archive.org/memento/20051028034156/http://www.data-archive.ac.uk/news/publications/UKDAPreservationPolicy0905.pdf, accessed on 1 November 2012

UK Data Archive (UKDA) (2011). Preservation Policy. Retrieved from http://www.data-archive.ac.uk/media/54776/ukda062-dps-preservationpolicy.pdf, accessed on 1 November 2012

van der Hoeven, J. (2007). "Dioscuri: emulator for digital preservation". D-Lib Magazine 13 (11/12). DOI:10.1045/november2007-inbrief. Retrieved from http://www.dlib.org/dlib/november07/11inbrief.html. Accessed on 3 August 2012

van der Knijff, J., Wilson, C. (2011). Evaluation of Characterisation Tools. Part 1: Identification. SCAPE Technical Report. September 2011. Retrieved from http://www.scape-project.eu/wp-content/uploads/2012/01/SCAPE_PC_WP1_identification21092011.pdf, accessed on 9 November 2012

W3C (2001). Web Services Description Language (WSDL) 1.1. Retrieved from www.w3.org/TR/wsdl, accessed on 9 November 2012

W3C (2011a). The PROV Data Model and Abstract Syntax Notation. W3C Working Draft 18 October 2011. Retrieved from http://www.w3.org/TR/2011/WD-prov-dm-20111018/, accessed on 10 August 2012

W3C (2011b). The PROV Ontology: Model and Formal Semantics. W3C Working Draft 13 December 2011. Retrieved from http://www.w3.org/TR/2011/WD-prov-o-20111213/, accessed on 10 August 2012

w3schools.com (nd). Introduction to XML Schema. Retrieved from http://www.w3schools.com/schema/schema_intro.asp, accessed on 2 November 2012

W3C OWL Working Group (2009). OWL 2 Web Ontology Language, Document Overview, W3C Recommendation 27 October 2009. Retrieved from http://www.w3.org/TR/owl2-overview/, accessed on 26 October 2012

Warner, J., Kleppe, A. (2003). The Object Constraint Language: Getting Your Models Ready for MDA (p. 240). Version 2.3.1., January 2012. Addison-Wesley Longman Publishing Co., Inc. Boston,

MA, USA. Retrieved from http://www.omg.org/spec/OCL/2.3.1/PDF/, accessed on 2 November 2012

Wilson, A. (2007). Significant Properties Report, InSPECT Work Package 2.2, Draft/Version 2. Retrieved from http://www.significantproperties.org.uk/wp22_significant_properties.pdf, accessed on 9 November 2012

Woods, K., Brown, G. (2008). Creating Virtual CD-Rom Collections. iPRES 2008: The Fifth International Conference on Preservation of Digital Objects Retrieved from http://www.bl.uk/ipres2008/ipres2008-proceedings.pdf, accessed on 9 November 2012

WSBPEL (2007). Execution Language Version 2.0. OASIS Standard. OASIS Web Services Business Process Execution Language (WSBPEL) Technical Committee. 11 April 2007. Retrieved from http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, accessed on 16 November 2012

Yeo, G. (2010). "Nothing is the same as something else": significant properties and notions of identity and originality. Archival Science , 10 (2) 85 - 116. DOI 10.1007/s10502-010-9119-9. Retrieved from http://discovery.ucl.ac.uk/1318039/3/1318039.pdf, accessed on 30 November 2012

Zierau, E., Kejser, U., Kulovits, H. (2010). Evaluation of Bit Preservation Strategies. iPRES 2010: The Seventh International Conference on Preservation of Digital Objects Retrieved from http://www.ifs.tuwien.ac.at/dp/ipres2010/papers/zierau-31.pdf, accessed on 9 November 2012

Copies of all top-level websites at the time of access can be obtained from the author.